View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Padua@thesis

DEPARTMENT OF INFORMATION ENGINEERING
UNIVERSITY OF PADOVA

Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea Magistrale in Ingegneria dell'Automazione

Tesi di Laurea Magistrale

Traffic Models for Data Networks and Energy Efficient Control Techniques

Candidato
Michielan Marco

Relatore
Prof. Cenedese Angelo

Correlatori
Prof. Vitturi Stefano
Ing. Tramarin Federico

Anno Accademico 2013-2014

“ All’ingegnere compete rivestire
di vita, conforto e speranza
lo scheletro della scienza.”

Herbert Hoover

Contents

Introduction **9**

1 An Overview of Traffic Models **11**

1.1 Preliminaries About Traffic Modeling 11

1.2 The Poisson and the Compound Poisson Traffic Models 12

1.2.1 Limitations of the Poisson model 12

1.3 The Packet Train Model 14

2 The Self-Similar Model **15**

2.1 Deterministic and Stochastic Self-Similarity 15

2.2 The Hurst Parameter 17

2.3 State of Art 18

2.4 Mathematical Background 19

2.4.1 Properties 20

2.4.2 Heavy-Tailed Distributions and Predictability 21

2.4.3 Heavy-Tailed Distributions and Long Range Dependence . . . 22

2.5 Inference for Self-Similar Processes 24

3 Energy Efficient Ethernet **27**

3.1 Introduction to the ISO/OSI Model 27
3.1.1 Description of OSI Layers 27
3.2 Introduction to Ethernet 30
3.3 Adaptive Link Rate 32
3.4 Low Power Idle Mode 34
3.4.1 Description of EEE working 35
3.4.2 Frame vs Burst Transmission Policy 35
3.5 Evaluation of EEE performances 36

4 Control for Self-Similar Network Traffic 39

4.1 Structural Causality 39
4.2 Predictability of Self-Similar Traffic 40
4.2.1 Estimation of Conditional Probability Density 41
4.2.2 Predictability and Time Scale 42
4.2.3 On-Line Estimation of Future Traffic 43
4.3 SAC and Underlying Congestion Control 45
4.3.1 Underlying Congestion Control 45
4.3.2 Selective Aggressiveness Control (SAC) 46
4.4 EEEP Strategy 48
4.5 Expected Results 52

5 Simulations Results 55

5.1 Artificial Generation of the Network Traffic 55
5.2 Real Traffic Data 56
5.2.1 AuckVIII Series 56

5.2.2 San Diego Series 57

5.3 Parameter tuning for EEEP 59

5.4 Simulations of EEEP Strategy on San Diego Series 64

Conclusions **73**

A Other simulations **75**

B Matlab Code **81**

Bibliography **103**

Introduction

The interest in studying communications networks, born some decades ago, mainly with vocal transmissions, has also involved the computer environment, in particular the field of data transmissions among nodes of a network. Until the '90s the research communities thought that data traffic could be adequately described by certain Markovian models. This supposition has been denied by the discovery and the recognition that actual data traffic is different from telephony traffic and this fact influenced the networking research landscape, necessitating a reexamination and a changing of some of its basic premises.

The goal of this thesis is first to present some models which can describe the behavior of traffic data in a network (a LAN for example) and second to explain some efficient techniques to save energy during the data transmission. In particular it will be presented a new technique which combines the statistic properties of the traffic and the great energy savings coming from the introduction of the recent standard IEEE 802.3az. In the last part of the thesis the results of the simulations obtained implementing the new algorithm proposed will be showed with interesting developments.

Chapter 1 presents some models that try to describe traffic data with particular attention to the most famous one, that is, the Poisson model with the conclusion that only a self-similar model can capture the bursty behavior of the traffic data.

Chapter 2 directs its attention toward the great topic of self-similarity: after some considerations about the state of art, some mathematical concepts with definitions are presented with the last part dedicated to the identification techniques for the Hurst parameter.

Chapter 3 deals with Energy Efficient Ethernet: the two main methods on which the standard IEEE 802.3az could put its bases are presented with a focus on the

description of the LPI mode working. Finally some results obtained analyzing a real Ethernet card with the introduction of this standard are reported.

Chapter 4 presents some useful tools for the control of data networks with the final part dedicated to the description of the control strategy proposed with a final flow chart which helps the visualization of the algorithm in all its steps.

Chapter 5 shows the results of the simulations obtained using the software MATLAB and in the particular the time and energy savings reached using the EEE strategy with the use of the prediction of future traffic.

Appendices report other simulations done and the MATLAB code used.

Chapter 1

An Overview of Traffic Models

The development of traffic models is an important research area in the field of the communication networks. The interest towards this area is high for mainly two reasons. First, finding a correct model that can accurately represent the characteristics of the traffic is necessary in network simulations because it can be used in order to study algorithms and protocols to be applied to real traffic, and to analyze how traffic reacts to particular network conditions. Second, a good model helps for a better understanding of the factors which characterizes the network traffic itself. In this chapter the main traffic models used in the past are presented to have a general overview on the topic which this thesis is going to analyze.

1.1 Preliminaries About Traffic Modeling

Data traffic can be modeled as a sequence of arrivals of discrete entities such as packets. There are two equivalent representations: *counting processes* and *inter arrival time processes*. A counting process $\{N(t)\}_{t=0..\infty}$ is a continuous-time, integer-valued stochastic process, where $N(t)$ expresses the number of arrivals in the time interval $(0, t]$. An inter arrival time process is a non-negative random sequence $\{A_n\}$ where $A_n = T_n - T_{n-1}$ indicates the length of the interval separating arrivals $n-1$ and n. In the case of *compound* traffic, arrivals may happen in *batches*, that is, several arrivals can happen at the same instant T_n. This fact can be modeled by using an additional non-negative random sequence $\{B_n\}_{n=1..\infty}$ where B_n is the cardinality of the n-th batch.

1.2 The Poisson and the Compound Poisson Traffic Models

The Poisson model is the oldest traffic model in use and it was introduced in the contest of telephony by A. K. Erlang. Traffic is characterized by assuming that the packet arrivals A_n are *independent* and that they are *exponentially distributed* with rate parameter λ where $P\{A_n \leq t\} = 1 - e^{-\lambda t}$.
The most important analytical properties of this kind of process are the following:

1. The superposition of independent Poisson processes with rates $\lambda_1, \lambda_2, ...\lambda_n$ results in a new Poisson process with rate $\lambda_1 + \lambda_2 + ... + \lambda_n$.
2. The number of arrivals in disjoint intervals is statistically independent (Poisson process is *memory less*).
3. Mean and variance are equal to λ.
4. The multiplexing of independent traffic streams approximates a Poisson process if: the traffic streams can be modeled as renewal processes (that is, inter arrival times are i.i.d.); as the number of streams increases the individual rates decrease so as to keep the aggregate rate constant.

In the compound Poisson model, the base Poisson model is extended to deliver *batches* of packets at once. The inter-batch arrival times are exponentially distributed while the batch size is geometric. Mathematically, this model has two parameters, λ, the arrival rate and ρ in $(0, 1)$, the batch parameter.
Thus, the mean number of packets in a batch is $\frac{1}{\rho}$, while the mean inter-batch arrival time is $\frac{1}{\lambda}$. Mean packet arrivals over time period T are $\frac{T\lambda}{\rho}$. The compound Poisson model shares some of the analytical benefits of the pure Poisson model: the model is still memory less and the aggregation of streams is still (compound) Poisson.

1.2.1 Limitations of the Poisson model

One basic limitation of the Poisson model is its inability to capture traffic burstiness which characterizes data traffic. A sequence of arrival times will be bursty if the T_n tend to form clusters, that is, if the corresponding $\{A_n\}$ sees a mix of relatively long and short inter arrival times. Mathematically speaking, traffic burstiness is related to short-terms autocorrelations between the inter arrival times. In any renewal traffic process the autocorrelation function of the $\{A_n\}$ vanishes identically while traffic

burstiness shows positive autocorrelation between the $\{A_n\}$. Thus, Poisson is not the appropriate model in case of bursty traffic, especially when traffic burstiness happens on multiple time scales.

Figure 1.1 (a)-(e) (taken from [**10**]) depicts a set of plots of the number of packets per time unit for different choices of time units. Starting with a time unit of 100 s (Fig.1.1(a)), each subsequent plot is obtained from the previous one by increasing the time resolution by a factor of 10. As it can be seen all plots are very "similar" to one another (in a distributional sense), that is, data traffic seems to look the same in the large (min, h) as in the small (s, ms). In particular at every time scale, bursts consist of bursty sub-periods separated by less bursty sub-periods.

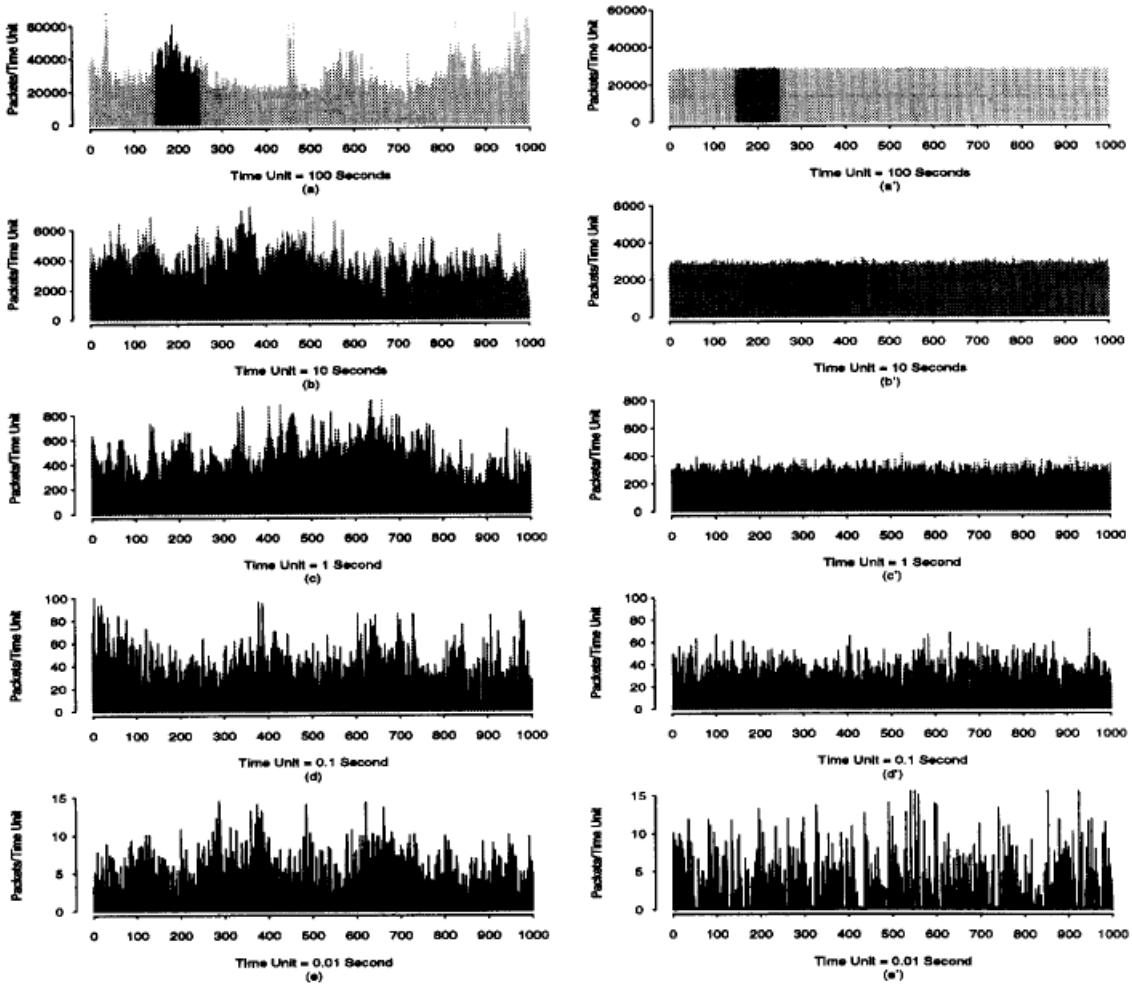

Figure 1.1: *Comparison between data traffic on five different time scales (a)-(e) and synthetic traffic from an appropriately chosen compound Poisson model on the same five scales (a')-(e').*

This behavior is very different from conventional telephone traffic: in fact this traffic typically produce plots of packet counts which are indistinguishable from white noise after aggregating over a few hundred milliseconds as illustrated in Fig.1.1 with the sequence of plots (a')-(e'); this sequence was obtained in the same way as the sequence (a)-(e), except that it depicts synthetic traffic generated from a comparable (in terms of average packet size and arrival rate) compound Poisson process. Fig.1.1 provides a simple method for distinguishing clearly between measured data and traffic generated by old classic models and strongly suggests the use of self-similar stochastic processes for traffic modeling purposes.

1.3 The Packet Train Model

This kind of model is a bit more complex than the Poisson one. In fact while the Poisson model is based on the hypothesis that packet arrivals are independent and unpredictable, the train model considers the dependence between packets moving between the same end-nodes. The motivation which supports this kind of model is the following. If packet arrivals are independent, then routing decision for packets moving between the same endpoints are performed independently on routers and this may lead to processing overhead. Otherwise if a train model is assumed, the routing decisions can be taken only when the head of a new packet train is detected and no overhead would incur on subsequent packets belonging to the same train. Another important fact motivates the use of this kind of model: in a network there is an upper bound for the dimension of files (in fact buffer sizes are bounded) and this fact determines the fragmentation of data in multiple small packets which however travel between the same endpoints and therefore they can be considered to belong to the same packet train.

For completeness, some other traffic models are here cited. There are the Renewal Traffic Models (Bernoulli processes), the Markov Traffic Models (Markov renewal, Markov Modulated) and the Auto-regressive Traffic Models. For a complete explanation it is sufficient to look at the huge amount of scientific literature on these fields (for example see [**1**] and [**23**]).

Chapter 2

The Self-Similar Model

As it was explained, the Poisson model is not able to capture traffic burstiness which characterizes data traffic. For this reason, a good starting point for a correct analysis of data traffic is the seminal study of Leland, Taqqu, Willinger and Wilson [**10**] which demonstrated that self-similarity is an important notion in the understanding of network traffic, also for the modeling and analysis of network performance. With their work, the authors considered an Ethernet LAN traffic: they demonstrated that this kind of traffic is statistically self-similar and that none of the used past models is able to capture this behavior which has serious implications for the design, control and analysis of these networks.

2.1 Deterministic and Stochastic Self-Similarity

Self-similarity and fractals are mathematical concepts studied mainly by Mandelbrot in [**12**]. An object can be called self-similar if some of its properties are preserved with respect to scaling in space and/or time. If an object is self-similar or fractal, its parts, when magnified, resemble the shape of the whole.
There are two kinds of self-similarity: deterministic self-similarity in which there is a strong form of recursive regularity and stochastic self-similarity, more general than the previous one. An example of deterministic self-similarity is the two dimensional (2D) Cantor set (see [**14**]) living on $A = [0, 1]$ x $[0, 1]$: it is obtained by starting with a solid or black unit square, scaling its size by $1/3$, then placing four copies of the scaled solid square at the four corners of A. If the same process of scaling followed by translation is applied recursively to the resulting objects ad infinitum, the limit set

thus reached defines the 2D Cantor set. This process is an example of deterministic self-similarity and it is illustrated in Figure 2.1.

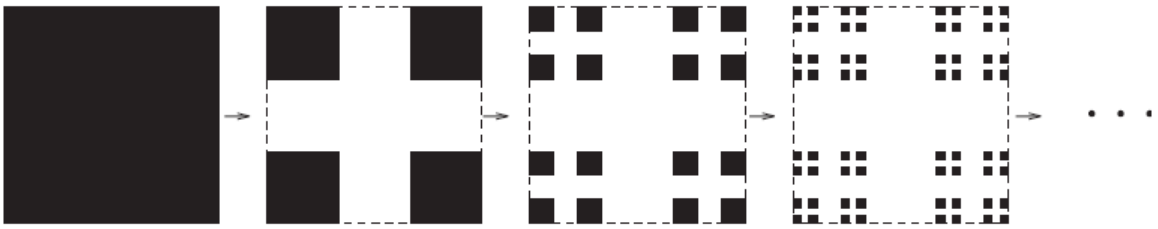

Figure 2.1: *2D-Cantor set.*

Stochastic self-similarity, instead, was introduced because it was necessary for traffic modeling purposes where stochastic variability is an essential component. Figure 2.2 (taken from [**14**]) shows a traffic trace where there is the plot of throughput against time with time granularity respectively 100 seconds (top left), 10 seconds (top right), 1 second (bottom left) and 0.1 seconds (bottom right). Unlike deterministic

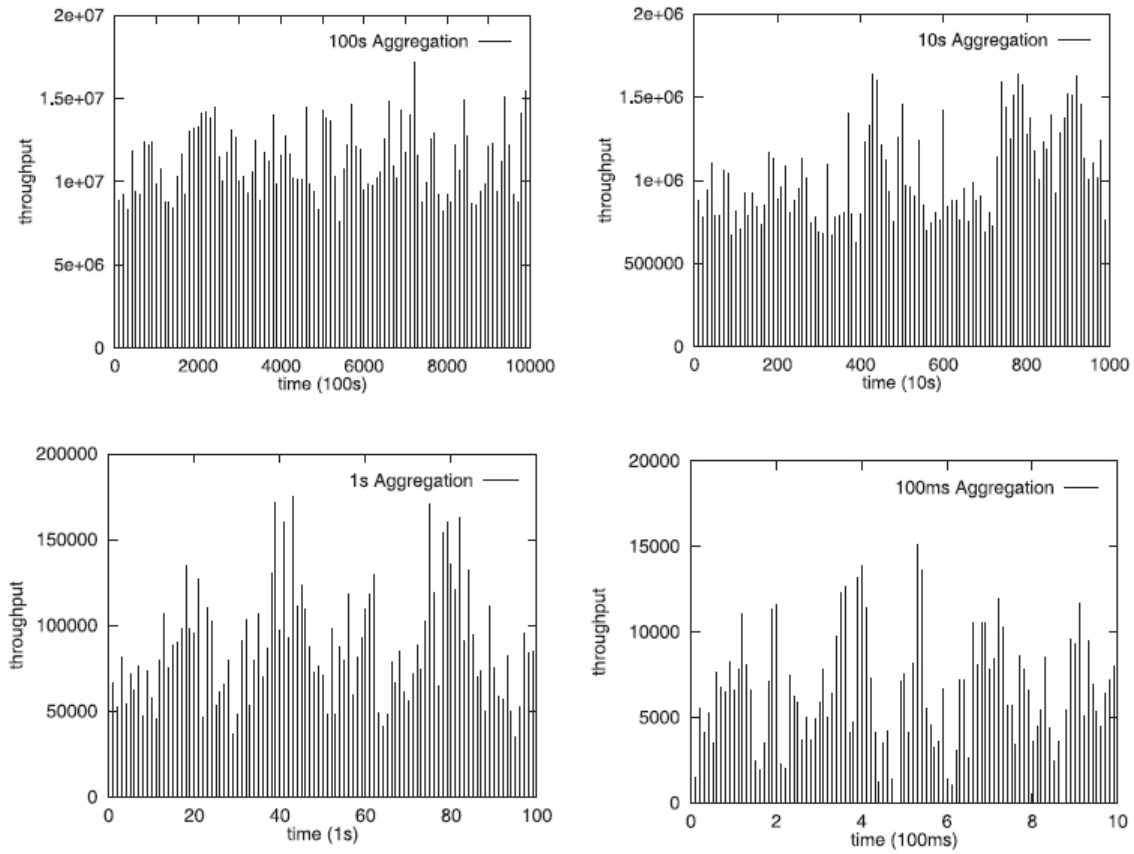

Figure 2.2: *Stochastic self-similarity across time scales.*

fractals, the objects in figure 2.2 do not possess exact resemblance of their parts

with the whole at finer details but, here, the assumption is that the measure of resemblance is the shape of a graph with the magnitude suitably normalized.

Moreover, traffic series can be considered as sample paths of stochastic processes. According to this point of view, second-order statistics of these processes are able to capture burstiness and variability. This idea, as it can be seen in the next paragraphs, will be useful to study data traffic using math properties.

2.2 The Hurst Parameter

This paragraph has the aim to introduce an important mathematical parameter that characterizes the stochastic processes of traffic series, that is, the Hurst parameter.

The Hurst parameter is defined in terms of the asymptotic behavior of the rescaled range as a function of the time span of a time series as follows:

$$E\Big[\frac{R(n)}{S(n)}\Big] = Cn^H \quad as \quad n \to \infty \tag{2.1}$$

where n is the time span of the observation (number of data points in a time series), $R(n)$ is the range of the first n values, $S(n)$ is their standard deviation and C is a constant.

The Hurst parameter is used as a measure of long term memory of time series and it relates to the autocorrelations and the rate at which these decrease as the lag between pairs of values increases. In fractal geometry, the Hurst parameter, has been denoted by H in honor of both Harold Edwin Hurst (1880-1978) and Ludwig Otto Hölder (1859-1973) by Benoît Mandelbrot (1924-2010).
The Hurst parameter is an index of long-range dependence. It quantifies the relative tendency of a time series either to regress strongly to the mean or to cluster in a direction (see [**25**]).

- A value H in the range $0.5 < H < 1$ indicates a time series with long-term positive autocorrelation, meaning both that a high value in the series will probably be followed by another high value and that the values a long time into the future will also tend to be high.

- A value H in the range $0 < H < 0.5$ indicates a time series with terms switching

between high and low values, meaning that a single high value of the time series will be probably followed by a low value and that the value after that will tend to be high with this tendecy to switch between high and low values lasting a long time into future.

- A value of $H = 0.5$ can indicate a completely uncorrelated series, that is, the process is short-range dependent.

- A value of $H = 1$ is uninteresting since it leads to a degenerate situation.

- A value of $H > 1$ is prohibited if the process is stationary.

2.3 State of Art

Previous works on self-similarity may be classified into four categories.

- **Measurement-based traffic modeling:** traffic traces from physical networks are collected and analyzed to detect, identify and quantify pertinent characteristics using also statistical inference techniques (Hurst parameter estimation). A significant step toward the development of accurate analysis is the introduction of wavelet-based techniques (see [**4**],[**6**],[**16**],[**22**]).

- **Physical modeling:** tries to find models of network traffic that relate to the physics of how traffic is generated in an actual network. It has been studied that in a network, hosts exchanging files whose sizes are heavy tailed (see next section), give rise to self-similar traffic. This fact comes from some studies that establish that the superposition of a large number of independent on/off sources with heavy-tailed on and/or off periods leads to self-similarity in the aggregated process whose long-range dependence is determined by the heavy tailedness of on or off periods (see [**15**],[**18**],[**22**]).

- **Queueing analysis:** there are works that provide mathematical models of long-range dependent traffic with a view toward facilitating performance analysis in the queueing theory sense: in fact they establish basic performance boundaries by investigating queueing behavior with long-range dependent input (see [**5**],[**11**],[**13**],[**19**]).

- **Traffic control:** has two subcategories: resource provisioning and dimensioning which can be viewed as a form of open-loop control, and closed-loop or feedback traffic control. Due to their feedback-free nature, the works on queueing analysis with self-similar input have direct bearing on the resource dimensioning problem. On the feedback control side there is the work on multiple time scale congestion control, which tries to exploit correlation structure that exists across multiple time scales in self-similar traffic for congestion control purposes. Long-range dependence admits the possibility of exploiting correlation at large time scales to build a predictability structure which, in turn, can be guide congestion control actions to obtain significant performance gains (see [**8**],[**20**],[**21**]).

2.4 Mathematical Background

With the idea that self-similar traffic can be studied using stochastic processes, this paragraph presents some important mathematical concepts on stochastic processes. Let $X(t)$ be a *covariance stationary* stochastic process (or also *second-order stationary* process) with mean $\mu := E[X(t)]$ (for simplicity, in the following paragraphs, $\mu = 0$), variance $\sigma^2 := E[(X(t) - \mu)^2]$ for all $t \in \mathbb{Z}$ and autocovariance function $\gamma(k), k \geq 0$. Let $X^{(m)}$ be the *aggregate process* of X at aggregation level m defined as:

$$X^{(m)}(i) := \frac{1}{m} \sum_{t=m(i-1)+1}^{mi} X(t).$$

For each m, $X(t)$ is partitioned into non-overlapping blocks of size m, their values are averaged and i is used to index these blocks. Let $\gamma^{(m)}(k)$ denote the autocovariance function of $X^{(m)}$. Under these assumptions, the following definitions can be given.

Definition (Second-Order Self-Similarity): $X(t)$ is *exactly second-order self-similar* with Hurst parameter H if

$$\gamma(k) = \frac{\sigma^2}{2}((k+1)^{2H} - 2k^{2H} + (k-1)^{2H}) \tag{2.2}$$

for all $k \geq 1$. $X(t)$ is *asymptotically second-order self-similar* if

$$\lim_{m \to \infty} \gamma^{(m)}(k) = \frac{\sigma^2}{2}((k+1)^{2H} - 2k^{2H} + (k-1)^{2H}). \tag{2.3}$$

It can be checked that Eq.2.2 implies $\gamma(k) = \gamma^{(m)}(k)$ for all $m \geq 1$. Thus, second-order self-similarity captures the property that the correlation structure is exactly or asymptotically preserved under time aggregation.

2.4.1 Properties

Mathematically, self-similarity manifests itself in a number of equivalent ways:

1. **(Slowly Decaying Variances)** The variance of the sample mean decreases more slowly than the reciprocal of the sample size, i.e., $var(X^{(m)}) \propto am^{-\beta}$ as $m \to \infty$, with $0 < \beta < 1$ and a is a finite positive constant;

2. **(Long Range Dependence)** The autocorrelations decay hyperbolically rather than exponentially fast, implying a non-summable autocorrelation function $\sum_{k=-\infty}^{\infty} r(k) = \infty$;

3. The ***spectral density*** $\Gamma(\nu) = (2\pi)^{-1} \sum_{k=-\infty}^{\infty} r(k)e^{ik\nu}$ satisfies the property $\Gamma(\nu) \propto c|\nu|^{-\alpha}$ for $\nu \to 0$, $c > 0$ and $0 < \alpha = 2H - 1 < 1$, that is, the spectral density diverges around the origin, implying ever larger contributions by low-frequency components.

As already explained, the value of the Hurst parameter H is very important because it characterizes the traffic itself. In fact, looking at the definition of self-similar process:

- if $H = \frac{1}{2}$ then $r(k) = 0$ and $X(t)$ is short-range dependent by the fact that its samples are completely uncorrelated.

- if $0 < H < \frac{1}{2}$ then $\sum_{k=-\infty}^{\infty} r(k) = 0$, a condition rarely encountered in applications.

- if $H = 1$ then $r(k) = 1$ for all $k \geq 1$ and this case is not interesting.

- $H > 1$ is prohibited due to the stationary condition on $X(t)$.

For these reasons, the range of value of interest is $\frac{1}{2} < H < 1$.

2.4.2 Heavy-Tailed Distributions and Predictability

There is a strict relationship between heavy-tailed distributions and long-range dependence which characterizes self-similarity. A random variable Z has a *heavy-tailed distribution* if

$$P[Z > x] \propto cx^{-\alpha}, x \to \infty \tag{2.4}$$

where $0 < \alpha < 2$ is called the *tail index* and c is a positive constant. That is, the tail of the distribution, asymptotically, decays hyperbolically. This is in contrast to *light-tailed distributions* as exponential or Gaussian distribution which possess an an exponentially decreasing tail. In the networking context, the case of interest is $1 < \alpha < 2$ and a frequently used heavy-tailed distribution is the *Pareto distribution* whose distribution is given by

$$P[Z \leq x] = 1 - \left(\frac{b}{x}\right)^{\alpha}, b \leq x \tag{2.5}$$

with α the tail index and b called the *location parameter*.

The main characteristic of a random variable obeying a heavy-tailed distribution is that it exhibits extreme variability: in fact it can assume large values with non negligible probability so that sampling from such a distribution gives a lot samples with "small" values but also it can give few samples with "very high" values.

The relationship among heavy-tailedness, long-range dependence and then self-similarity is related to the heavy-tailedness of some network variables, as file sizes and connection duration, which is the main cause of long-range dependence and self-similarity in network traffic.

A simple example (see [**14**]) of the intrinsic predictability associated with heavy-tailed random variables is presented here. Let Z be a heavy-tailed random variable interpreted as the *duration* of a network connection. The assumption is that a connection has been active for $\tau > 0$ seconds and to simplify the discussion, time is considered discrete. The question could be: what is the probability that the connection will persist into the future given that it has been active for τ seconds? This probability can be expressed as:

$$\mathcal{L}(\tau) = 1 - \frac{P[Z = \tau]}{P[Z \geq \tau]}$$

For light tails, in particular distributions with asymptotically exponential tails

$P[Z > x] \propto c_1 e^{-c_2 x}$ where c_1, c_2 are constants, this probability, for large τ, is $\mathcal{L}(\tau) \propto e^{-c_2}$. Thus prediction is not increased by conditioning on ever longer periods of observed activity. For heavy tails, instead, this probability $\mathcal{L}(\tau) \to 1$ for $\tau \to \infty$. Thus the longer the period of observed activity, the more certain that it will persist into the future. In conclusion, in the heavy-tailed case, by conditioning the prediction on a sufficiently long past observation of activity, the prediction error can be reduced to an arbitrary small level.

2.4.3 Heavy-Tailed Distributions and Long Range Dependence

As seen, heavy tails lead to predictability and, therefore, to long-range dependence in network traffic. Now other definitions are introduced:

Definition(FBM):
$Y(t)$ is called *fractional Brownian motion* with parameter H, $0 < H < 1$, if $Y(t)$ is Gaussian and H-sssi (that is, self-similar with increments $X(t) := Y(t) - Y(t-1)$ stationary).

Definition(FGN):
$X(t)$ is called *fractional Gaussian noise* with parameter H if $X(t)$ is the increment process of FBM with parameter H.

These kind of processes are Gaussian self-similar processes with long-range dependence. Their Gaussian structure renders them useful as aggregate traffic models where aggregation of independent traffic sources leads to the Gaussian property.

After these definitions, it could be examined why heavy tails are considered the root cause of long-range dependence in network traffic considering a constructive approach (see also [**14**]).
The on/off model considers N independent traffic sources $X_i(t)$, $i \in [1, N]$, where each is 0/1 *reward renewal process* with i.i.d. on periods and i.i.d. off periods. This just means that $X_i(t)$ takes on the values 1(on periods) and 0 (off periods) on alternating, non-overlapping time intervals called on and off periods, respectively. $X_i(t) = 1$ is interpreted as there being a packet transmission. Three such on/off sources and their aggregation are depicted in Fig.2.3.

Let $S_N(t) = \sum_{i=1}^{N} X_i(t)$ denote the aggregate traffic at time t and let $Y_N(Tt)$ the cumulative process defined as

$$Y_N(Tt) = \int_0^{Tt} \Big(\sum_{i=1}^{N} X_i(s) \Big) ds$$

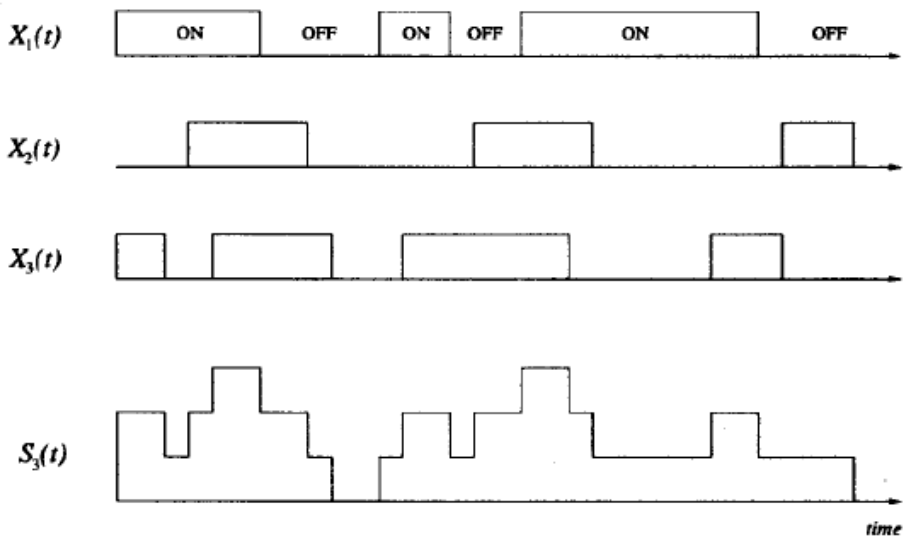

Figure 2.3: *Three on/off sources $X_1(t), X_2(t), X_3(t)$ and their aggregation $S_3(t)$.*

where $T > 0$ is a scale factor. Thus $Y_N(Tt)$ measures the total traffic up to time Tt. How does heavy-tailedness influence long-range dependence? Let τ_{on} be the random variable describing the duration of the on periods and let τ_{off} be the random variable associated with the durations of the off periods. Let

$$P[\tau_{on} > x] \propto cx^{-\alpha}, \quad x \to \infty$$

where $1 < \alpha < 2$ and $c > 0$ is a constant. As to τ_{off}, it can be either heavy tailed or light tailed with finite variance. It can be shown that $Y_N(Tt)$ behaves like FBM in the following sense.

Theorem (On/Off Model and FBM): $Y_N(Tt)$ behaves statistically as

$$\frac{E(\tau_{on})}{E(\tau_{on}) + E(\tau_{off})} NTt + CN^{1/2}T^H B_H(t) \tag{2.6}$$

for large T, N where $H = (3 - \alpha)/2$, $B_H(t)$ is FBM with parameter H and $C > 0$ is a quantity depending only on the distributions of τ_{on} and τ_{off}.

Thus $Y_N(Tt)$ asymptotically behaves as fractional Brownian motion fluctuating around $NTt\frac{E(\tau_{on})}{E(\tau_{on})+E(\tau_{off})}$ when suitably normalized. It is long-range dependent ($\frac{1}{2} < H < 1$) iff $1 < \alpha < 2$; that is, τ_{on}'s distribution is heavy-tailed. If neither τ_{on} nor τ_{off} is heavy-tailed, then $Y_N(Tt)$ is short-range dependent. It is in this sense that heavy-tailedness (of the on or off periods) is an essential component to inducing long-range dependence in the aggregated time series.

2.5 Inference for Self-Similar Processes

As already seen, a covariance stationary self-similar process manifests itself in three different ways:

1. Slowly decaying variances

2. Long-range dependence

3. A spectral density which diverges around the origin

It is now interesting to study the problem of estimating the degree of self-similarity H because it is able to completely describe the dynamic of a self-similar process.

There are mainly three simple techniques for the identification of H:

- **R/S-statistic**

  it is a technique in the time domain which exploits the so-called Hurst effect. For a given set of observations $(X_k, k = 1, 2, ..., n)$ with sample mean $\bar{X}(n)$ and sample variance $S^2(n)$, the *rescaled adjusted range statistic* (or *R/S statistic*) is given by $R(n)/S(n) = 1/S(n)[max(0, W_1, W_2, ..., W_n) - min(0, W_1, W_2, ..., W_n)]$, with $W_k = (X_1 + X_2 + ... + X_k) - k\bar{X}(n)$ and $k \geq 1$.
  While many naturally occurring time series appear to be well represented by the relation $E[R(n)/S(n)] \sim an^H$ as $n \to \infty$, with Hurst parameter typically about 0.7, observations X_k from a short-range dependent model are known to satisfy $E[R(n)/S(n)] \sim bn^{0.5}$ as $n \to \infty$. This discrepancy is referred to as the *Hurst effect*.
  Coming back to the R/S analysis, this technique consists of taking logarith-

mically spaced values n and plotting $log(R(n)/S(n))$ versus $log(n)$ results in the *pox diagram of R/S*. After an initial transient zone, the graph can be approximated with a straight line of a certain slope which, if it is correct, takes value between 1/2 and 1. This slope is the estimate $\hat{H}$ of H.

- **Analysis of the variances**

this technique considers the variances of the aggregated processes $X^{(m)}$. As it can be shown, these variances decrease linearly (for large m) in log-log plots against m with slopes arbitrarily flatter than -1. The so-called *variance-time plots* are obtained by plotting $log(var(X^{(m)}))$ against $log(m)$ and by fitting a simple least squares line through the resulting points in the plane, ignoring the small values for m. Values of $\hat{\beta}$ of the asymptotic slope between -1 and 0 give an estimate for the degree of self-similarity $\hat{H} = 1 + \hat{\beta}/2$.

- **Periodogram-based analysis**

it is a technique in the frequency domain. It is more refined than the two previous ones because it combines maximum likelihood estimates and the so-called *periodogram* $I(x) = (2\pi n)^{-1} |\sum_{j=1}^{n} X_j e^{ijx}|^2$, $0 \leq x \leq \pi$ of $X = (X_1, X_2, ..., X_n)$ and its distributional properties.
In particular, for Gaussian processes, Whittle's approximate MLE approach and the aggregation method discussed earlier, combined, give rise to an operational procedure for obtaining confidence intervals for the self-similarity parameter H.
Briefly, given time series, for each of the aggregated series, this method estimates the self-similarity parameter $H^{(m)}$ via Whittle's method and gives the corresponding 95%-confidence intervals of the form $\hat{H}^{(m)} \pm 1.96\hat{\sigma}_{H^{(m)}}$ where $\sigma^2_{H^{(m)}}$ is given by a known central limit theorem result. Plots of $H^{(m)}$ versus m will typically vary for small aggregation levels but will stabilize after a while and fluctuate around a constant value, the final estimate of H.

Fig.2.4 (taken from [**10**]) reports the graphics of the methods above described:

- The first figure refers to the R/S analysis in which the slope of the line con-

necting the points takes values between 1/2 and 1.

- The second figure refers to the variance analysis with slopes of the line between -1 and 0.

- The third figure is the periodogram analysis.

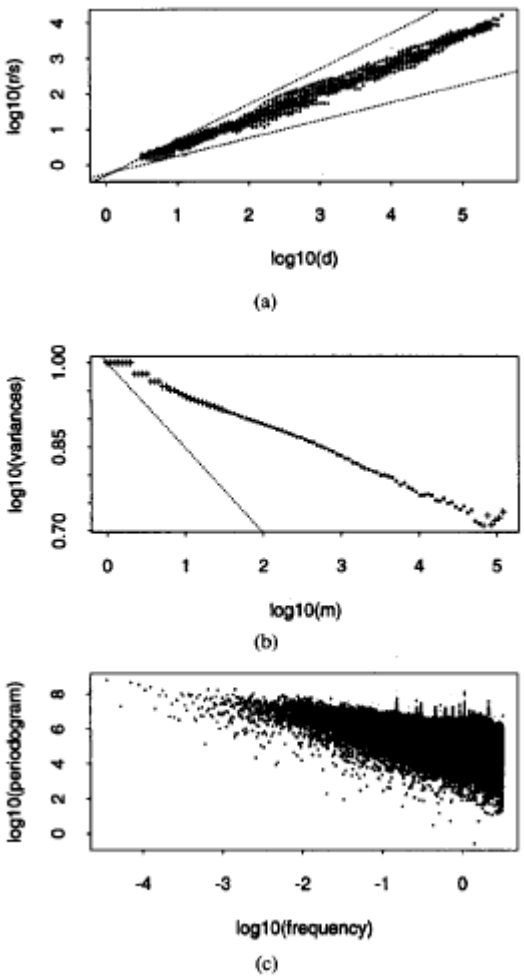

Figure 2.4: *Graphical examples of the methods described to find H. (a) is a R/S analysis, (b) is a variance analysis and (c) is the periodogram plot.*

Chapter 3

Energy Efficient Ethernet

3.1 Introduction to the ISO/OSI Model

The Open Systems Interconnection (OSI) model (ISO/IEC 7498-1) is a conceptual model that characterizes and standardizes the internal functions of a communication system by partitioning it into abstraction layers. The model is a product of the Open Systems Interconnection project at the International Organization for Standardization (ISO).

The model groups similar communication functions into one of seven logical layers. A layer serves the layer above it and is served by the layer below it. For example, a layer that provides error-free communications across a network provides the path needed by applications above it, while it calls the next lower layer to send and receive packets that make up the contents of that path. Two instances at one layer are connected by a horizontal connection on that layer.

3.1.1 Description of OSI Layers

OSI model has seven layers, labeled 1 to 7, with layer 1 at the bottom. Each layer is generically known as an N layer and an "N entity" (at layer N) requests services from an "N-1 entity" (at layer N-1). The dialog between the two entities carried out via an exchange of service messages called primitives and data messages called packet data unit (PDU) through the service access point (SAP). In this way, a layer-N

entity of system A gets to talk a peer layer-N entity of system B, as messages get down on system A and then up on system B. When a layer-(N+1) PDU has to be sent, it passes down through the layer-N SAP. The N layer does not simply carry this PDU to the underlying layer but has to add some information needed for the layer-N protocol to work. As a consequence, a protocol control information (PCI) is put before the layer-(N+1) PDU (which in the context of layer-N is called layer-N service data unit (SDU) to highlight that it is coming from the SAP) to build the layer-N PDU. As said before, the service interaction through SAPs is carried out through the exchange of primitives which usually are of four types:

1. Request: a primitive sent by layer (N+1) to layer N to request a service.

2. Confirm: a primitive returned to layer (N+1) from layer N to advise of activation of a requested service.

3. Indication: a primitive provided by layer (N+1) in reply to an indication primitive.

4. Response: a primitive returned to the requesting (N+1)st layer by the Nth layer to acknowledge or complete an action previously invoked by a request primitive.

In Fig.3.1 there is a graphical representation of what it was explained.

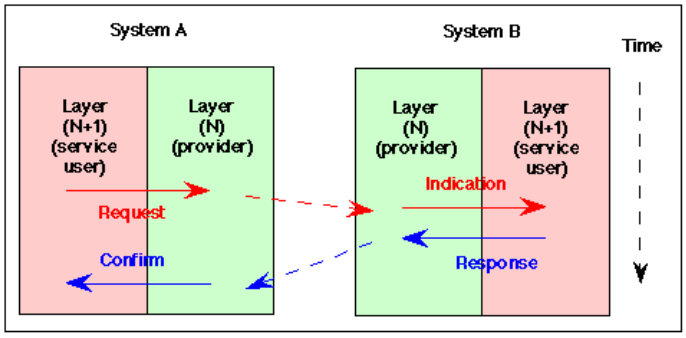

Figure 3.1: *Primitives for communications between peer protocol entities.*

In the following a short description of each layer is given (see [**24**]).

1. The *physical layer* is responsible for the actual delivery of the information over physical medium; for example, at the transmitter's side, by employing a modulator, a given waveform is sent over the medium when the bit is 0 and a different one is sent when the bit is 1; therefore, at the receiver's side,

a demodulator can reconstruct the 0/1 sequence based on the received waveform.

2. The *data link layer* is mainly responsible for enabling the local transfer of information. Its intervention is necessary in order to enable the actual exchange of bits through the physical layer. In this layer there are two sub-layers: the medium access control (MAC) which manages the presence of multiple users on the same medium and the logical link control (LLC) which provides the interaction between the upper layers and the MAC as well as performing flow and error control on a local basis.

3. The *network layer* is responsible for setting up, maintaining and closing the connection between the network layer entities of different subsystems. It makes it invisible to the upper layers how the network resources are utilized: the network layer entity on a system A does not know whether the flow of information is going through the systems B or C while exchanging information with a system D. To achieve this goal, the network layer has the responsibility of performing host addressing, that is, being able to identify all the nodes in the network in a unique manner, and routing, that is, to identify the desirable path the data has to follow throughout the network. Therefore, while the data link layer is concerned with local communication exchanges (it considers single links), the network layer has a global view (it considers the whole network).

4. The *transport layer* is responsible for creating a transparent virtual pipe between the two ends of the communication flow. Communication proceeds as though only two entities existed in the network; when the communication session is established the transport layer sets up this virtual pipe according to the requests of the session layer. It may include features to deliver an error-free communication, requesting the network layer to retransmit the pieces of information which are in error. Finally the transport layer can perform flow control, that is, it prevents the transport layer on one side from sending information faster than the rate that the receiving transport layer on other side can handle.

5. The *session layer* is responsible for establishing, suspending and tearing down in an orderly fashion any communication session the presentation layer might need.

6. The *presentation layer* is responsible for providing directly to the application

layer the session layer services as requested by the applications and for processing the data to make them usable to the application: the encryption and decryption of the data are usually made by this layer.

7. The *application layer* is the closest layer to the end user and it interacts with software applications that implement a communicating component. The functions of this layer typically include identifying communication partners, determining resource availability and synchronizing communication.

In Fig. 3.2 there is a graphical representation of the ISO/OSI model.

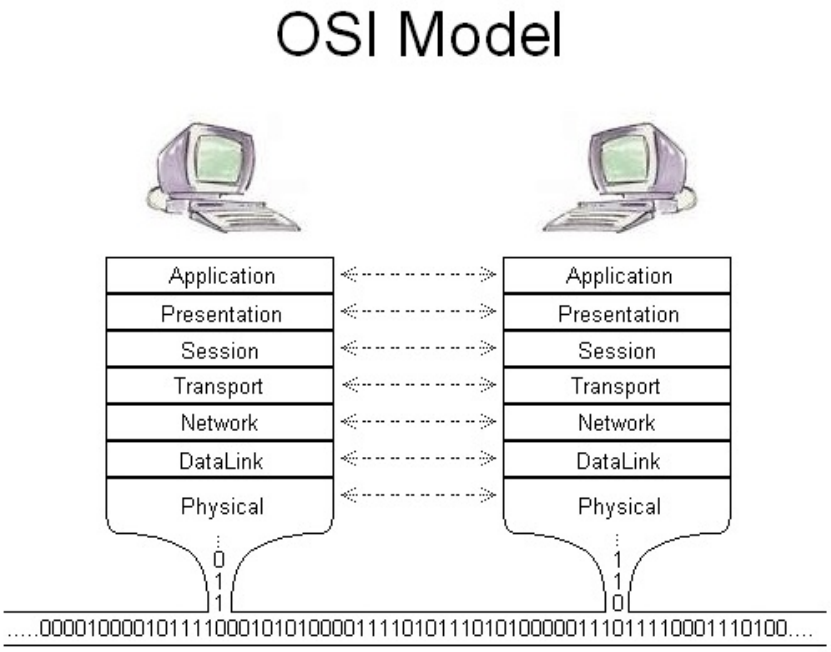

Figure 3.2: *ISO/OSI model.*

3.2 Introduction to Ethernet

Data network traffic which will be considered in the chapter dedicated to the simulations comes from an Ethernet LAN. Ethernet is a family of computer networking technologies for local area networks (LANs), commercially introduced in 1980, standardized in 1985 as IEEE 802.3 and it has largely replaced competing wired LAN technologies such as token ring, FDDI and ARCNET. The original 10BASE5 Ether-

net used coaxial cable as a shared medium but later the coaxial cables were replaced with twisted pair and fiber optic links in conjuction with hubs or switches.

Systems communicating over Ethernet divide a stream of data into shorter pieces called frames. The elements which form a frame are:

- *Preamble* (7 byte): these bytes wake up the receiving adapters and synchronize the oscillators with those of the sender.

- *Start Frame Delimiter (SFD)* (1 byte): it value is 10101011 and the series of the two bits to 1 indicates to the receiver that some important content is arriving.

- *Destination MAC address* (6 byte): this field contains the address of the destination LAN adapter; if the address does not match, the physical layer of the protocol discards it and sends it to the successive layers.

- *Source MAC address* (6 byte): this field contains the address of the source LAN adapter;

- *EtherLength (or EtherType)* (2 byte): this field contains the length of the payload.

- *Payload* (from 46 to 1500 byte): this field contains the real data: if the data exceed the maximum capacity, they are split into multiple packets while if the data do not reach the minimum length of 46 bytes, padding is added.

- *Frame Check Sequence (FCS)* (4 byte): it is used to detect any corruption of data in transit.

In Fig. 3.3 there is a graphical representation of an Ethernet frame.

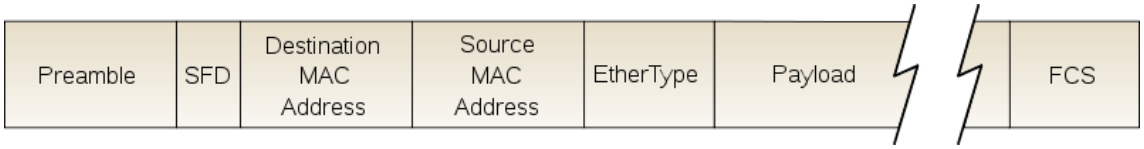

Figure 3.3: *Ethernet frame.*

Ethernet is one of the most important used wired communications technology in all the world. Due to the increasing request of energy, the topic about the energy consumption is becoming more and more relevant also for the field of networked systems. Therefore the problem of how to save energy also in the contest of Ethernet transmissions for LANs is very interesting, challenging and important. Ethernet is a technology widely used in private and public buildings: almost all produced computers have an Ethernet connection or more than one. There are mainly four kind of data rates supported in Ethernet using an UTP (Unshielded Twisted Pair) connection: 10 Mb/s (10BASE-T), 100 Mb/s (100BASE-TX), 1 Gb/s (1000BASE-T), and 10 Gb/s (10GBASE-T) and these data rates depend on how the UTP cable is used. One of the main characteristics of Ethernet is that for 100 Mb/s and higher rates, there is always transmissions among the nodes of the network because, also when there is no data to transmit, another signal called IDLE is sent to keep transmitters and receivers aligned. This property of Ethernet determines the fact that the energy consumption is very large. Typically, there is a consumption over 0.5 W for a 1000BASE-T Ethernet physical layer transceiver and over 5 W for a 10GBASE-T one. Considering these facts, in 2006 the IEEE 802.3 Working Group started to think about how to achieve energetic efficiency on Ethernet. After some years of hard work, in 2010 this effort gives life to IEEE 802.3az Energy Efficient Ethernet standard.

During the building of the Standard IEEE 802.3az, several technical methods upon which the standard had to put its bases were proposed and in particular two: ALR and LPI. Before considering LPI, which was selected by the standard, it is worth describing ALR (Adaptive Link Rate).

3.3 Adaptive Link Rate

This technique considers the opportunity to reduce link rate when the link itself is not much utilized. The goal is to maximize the time spent in a low data rate to obtain energy savings and to minimize the packet delay which could be introduced.

Here a brief description of the idea which underlies ALR policy is presented. First, it is necessary a mechanism for initiating and agreeing upon a link data rate change because either end of a link (a PC or a switch port) must be able to initiate a request to change the rate and there must be an agreement between the two ends of the link. This agreement, called also handshake, could be implemented in this way. When

one end of the link decides to increase or decrease its data rate, it sends an ALR REQUEST MAC frame to the other end; the receiving link partner sends an ALR ACK or an ALR NACK frame back if it respectively agrees or does not agree with the change of the data rate. An ALR ACK message changes the data rate and activates a link resynchronization.

In Fig.3.4 (taken from [**3**]) there is an illustration of how ALR method works.

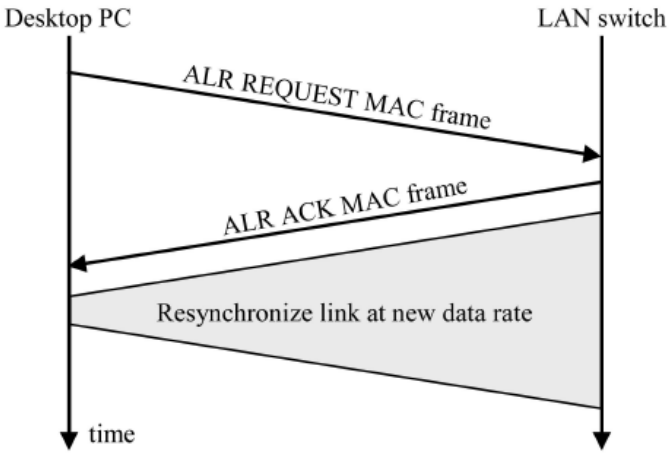

Figure 3.4: *ALR working.*

The policy that determines the change of the data rate is called Dual-Threshold Policy, based on the output queue length threshold crossing. This policy considers two thresholds to avoid undesirable oscillation between rates (hysteresis). If the output queue length exceeds the high threshold, then there is the request to increase the data rate and the only possible response from the other side is an ACK (it cannot disagree). Instead, if the queue length goes down under the low threshold, the request of reducing the data rate can be accepted or not and in this latter case the end which does not agree sends a NACK back. Now it is this end that must request the low data rate when its queue length is under the low threshold.

The Dual-Threshold Policy can oscillate between different data rates: in fact this oscillation will occur if the packet arrival at the low link data rate is high enough to cause a high threshold crossing at the low data rate but not high enough to maintain the queue length above the low threshold at the high data rate. To study this problem, an experiment was made using a card with link data rates of 100 Mbps and 1 Gbps given Poisson arrivals of packets of constant length of 1500 bytes. This system can be trivially modeled as an M/D/1 queue. Some observations can be done: as the packet arrival rate increases when in the low link data rate, the packet

delay also increases reaching an unacceptable level. Thus at some utilization level less than 10 percent of 1 Gbps, the data rate should be switched from 100 Mbps to 1 Gbps and remain in the 1 Gbps high link data rate.

To prevent oscillation, a utilization-threshold policy was developed. If link utilization is explicitly monitored, the effect of oscillations on packet delay can be reduced or eliminated. This new policy of utilization monitoring is based on counting the bytes sent in a certain interval: this further control permits to avoid all undesirable oscillations (see [**3**] for all the details).

Of course counting the number of bytes transmitted within a certain time requires additional accumulators and registers that could increase the complexity of the card which implements this strategy. Therefore a new heuristic policy, called ALR time-out-threshold policy, was designed. Two new timers are defined: the first one for the minimum time for the link to stay in the high data rate and the second one for the minimum time the link should stay in the low data rate before switching to the high data rate. The first timer is reset and restarted upon switching to the high rate and the link data rate is maintained at the high rate until the timer expires, irrespective of the queue length. When this timer has expired, if the queue length is under the low threshold, the link data rate is switched to the low rate. The second timer is always reset and restarted upon switching to the low data rate. Switching to the high rate is triggered by a queue threshold crossing of the high threshold even if the timer has not expired. This policy can be made adaptive: the initial value of the first timer can be doubled if the second timer has not expired when the high threshold is crossed. For the Markov model of the dual-threshold policy and other results see [**3**].

3.4 Low Power Idle Mode

As already anticipated, the IEEE 802.3az standard has the function to reduce power consumption in an Ethernet network. This standard describes a new state in the PHY (physical layer), called *low power idle* (LPI) mode which is enabled to consume only a very little quantity of power compared to the active normal state. Since Ethernet cards are not able to send or receive traffic when they are in LPI, the standard also specifies a protocol for treating the transitions from/to the LPI and the normal modes. What is interesting and challenging is the fact that the standard

does not specify the modes and the conditions upon which a network card should enter and leave LPI and this determined a great effort of the researchers to find the smartest way to use LPI and to save the most relevant quantity of energy.

Anyway basic EEE policy is currently the following: put Ethernet cards into the sleeping state when there is no transmissions and awake them if new data arrive.

3.4.1 Description of EEE working

Fig.3.5 (taken from [**9**]) shows the EEE working. Assuming that there is a link between two nodes of the network and that it is active, when there are no frames to transmit, the link turns off in the time t_s, enter the LPI mode and stays in this state (quiet state) until there is one frame (frame transmission policy) or some frames (burst transmission policy) to transmit. Therefore the link awakes in the time t_w and comes back to the active state, ready for the transmission. As it can be seen in the figure, there is also a periodic transmission of a refresh signal of length t_r to ensure that the receiver parameters are always aligned with the channel conditions.

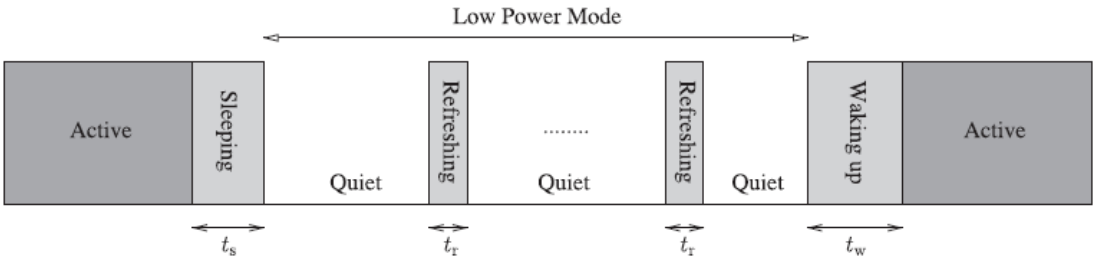

Figure 3.5: *EEE working using LPI.*

3.4.2 Frame vs Burst Transmission Policy

Here two of the most efficient and simplest algorithms to switch the PHYs into/out of the LPI mode are presented. The first is based on the *frame transmission*, that is, when a new frame has to be transmitted the network card awakens independently of the time elapsed in the LPI mode. The great benefit of this method is that it adds very low delay to the traffic because a transmission queue is not built while the interface stays in the LPI mode. From another point of view, this technique of

transmission has a great drawback: the potential energy savings could be drastically reduced due to the possible frequent changes between the active and the inactive power modes of the network. In fact the frame transmission method does not care about the number of transitions and in the worst case, there could be one transition for packet.

This problem is tackled by the second algorithm based on the *burst transmission* which awakes the device only after that a few data accumulate in the queue of the card. This reduces the number of transitions and improves energy efficiency but introduces another problem which must be considered, that is, the fact that there is an increase of the delay of the packets transmission. However the latter algorithm is one of the most promising ones because it is able to provide great energy savings with very low computational complexity and bounded delay.

3.5 Evaluation of EEE performances

This paragraph briefly reports the main results obtained in the article [**17**] which analyzes the performances of a NIC (Network Interface Card) that implements Energy Efficient Ethernet.

The goal is to show how EEE effectively works in a real Ethernet network. In this work the authors studied the performances of two NICs connected to the computer. A NIC is shown in Fig.3.6.

Figure 3.6: *NIC used in the experiments.*

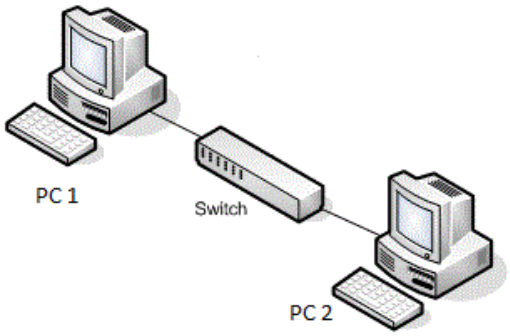

Figure 3.7: *Experimental setting.*

The experimental setting is represented in Fig.3.7. There are two computers: a sender (PC 1) and a receiver (PC 2) with a switch (with the NIC inside) that links the computers. The third experiments done refer to the measures of traffic made on the link between the switch and the PC 2.

The first experiment tried to study power consumption for two limit cases: no traffic on the link and a link with full load. Table 3.1 sums up the results.

	No EEE	EEE	No EEE	EEE
100BASE-TX	208	139	215	208
1000BASE-T	525	152	541	535

Table 3.1: *Measured NIC Power Consumption [mW] in the two cases: no traffic (first and second columns) and full load (third and fourth columns).*

As it can be observed from the Tab.3.1, EEE gives its benefits: in fact, when there is no traffic, the NIC power consumption is reduced by over 70% for 1 Gb/s and by over 30% for 100 Mb/s. This is the first important result that demonstrates the effectiveness of EEE standard.

The second experiment tried to estimate the power consumptions during mode transitions. Limiting the data rate to 10 Mb/s, the link load was low (1% at 1Gb/s and 10% at 100 Mb/s), packets are spaced by 200 μs and transitions between active and low power occurred frequently. This artificial way of generating and transmitting Ethernet traffic allowed the authors to evaluate the effects of the transitions, which are important to have a complete computation of the power consumptions. Table

3.2 sums up the results of the experiment.

	No EEE	EEE
100BASE-TX	215	201
1000BASE-T	531	512

Table 3.2: *Measured NIC Power Consumption [mW] when sending 5000 250 Byte Packets per Second.*

As it can be observed from the table, the power consumption in the two cases (with EEE and without EEE) is nearly the same. This is another significant result: the power consumption during transitions is close to that of the active mode. Therefore, if packets arrive spaced in time a large number of transitions will occur leading to a large energy consumption (see [**17**]).

The third experiment tried to measure the power consumption versus the link traffic load. The authors discovered that when the load reached 6%, the power consumption with and without EEE were similar. This means that in many practical applications, where the load is low, the transition overhead in EEE may be significant.

In summary, from the results obtained in the experiments, the authors extracted the following conclusions (here reported verbatim):

- The use of EEE will significantly reduce energy consumption when the NIC is in low power mode. In the experiments the reduction was more than 70% for 1 Gb/s.

- The energy consumption during EEE mode transitions is close to that of the active mode.

- The energy overhead caused by EEE transitions can be very significant at low loads.

From these observations, researchers can direct their studies towards mainly two directions in order to maximize energy savings: first trying to reduce the high power consumption during mode transitions and second trying to use the burst transmission as proposed in [**2**].

Chapter 4

Control for Self-Similar Network Traffic

As it was shown in the first two chapters, LAN traffic is very bursty at every time scale considered. Such behavior is very different from traditional models of network traffic, which show burstiness at short time scales but are smooth at large time scales because they have not the intrinsic property of long-range dependence. From a point of view, self-similarity has a bad effect on network performance, leading to increased delay and packet loss rate but from another one, its intrinsic long-range dependence could be exploited for control purposes. In fact, this characteristic of this kind of traffic implies the existence of a correlation structure, very useful for control aims. There is, indeed, the possibility of predicting how much traffic is expected in the network with sufficient reliability as it will be explained in the following paragraphs.

4.1 Structural Causality

As already mentioned in Chapter 2, there is a linear relationship between the tail index α of the Pareto distribution which characterizes the files sizes exchanged in the network and the Hurst parameter. In fact, the aggregate traffic that is induced by hosts exchanging files with heavy-tailed sizes over a network environment is self-similar, being more bursty the more heavy-tailed the file size distributions are (see [**14**]). This relationship between the observed traffic pattern and the property of the exchanged object sizes is called *structural causality*. Using an on/off model (a 0/1

renewal process with heavy-tailed on or off periods) and assuming independent traffic sources with no interactions, the relationship between α and H is the following:

$$H = \frac{3-\alpha}{2} \tag{4.1}$$

4.2 Predictability of Self-Similar Traffic

This paragraph presents a time domain method which exploits correlation structure in long-range dependent traffic to predict the future. This technique is based on the conditional expectation estimation (for all the details see [**14**]).

Given a wide-sense stationary stochastic process ξ_t with $t \in \mathbb{Z}$ and two positive numbers T_1 and T_2, it can be computed the quantity $a := \sum_{i\in[t-T_1,t)} q_i$ where q_i is a sample path of ξ_t over time interval $[t-T_1, t)$. Then let

$$V_1 := \sum_{i\in[t-T_1,t)} \xi_i, \qquad V_2 := \sum_{i\in[t,t+T_2)} \xi_i \tag{4.2}$$

where V_1 and V_2 are random variables that characterize the recent past and the near future. It could be interesting in computing the conditional probability

$$Pr[V_2 = b | V_1 = a] \tag{4.3}$$

where b is in the range of V_2. For example, if a represented a high traffic volume, then it could be interesting to know what the probability of encountering again another high traffic volume in the near future would be. Let

$$V^t_{max} := max \sum_{i\in[t-T_1,t)} q_i, \qquad V^t_{min} := min \sum_{i\in[t-T_1,t)} q_i \tag{4.4}$$

where V^t_{max} and V^t_{min} denote the highest and the lowest traffic volume seen so far at time t.

Now the range between V^t_{min} and V^t_{max} has to be partitioned into h *levels* with quantization step $\mu = (V^t_{max} - V^t_{min})/h$:

$$(0, V^t_{min}+\mu), [V^t_{min}+\mu, V^t_{min}+2\mu), [V^t_{min}+2\mu, V^t_{min}+3\mu)...$$
$$[V^t_{min}+(h-2)\mu, V^t_{min}+(h-1)\mu), [V^t_{min}+(h-1)\mu, \infty).$$

Two new random variables L_1 and L_2 are now defined:

$$\begin{aligned} L_k = 1 &\Leftrightarrow V_k \in (0, V^t_{min} + \mu) \\ L_k = 2 &\Leftrightarrow V_k \in [V^t_{min} + \mu, V^t_{min} + 2\mu) \\ &\vdots \\ L_k = h-1 &\Leftrightarrow V_k \in [V^t_{min} + (h-2)\mu, V^t_{min} + (h-1)\mu) \\ L_k = h &\Leftrightarrow V_k \in [V^t_{min} + (h-1)\mu, \infty) \end{aligned}$$

This is like a quantization. In fact, if L_k is near to 1, then the traffic level is low, while if L_k is near to h, the traffic level is high.
With these new definitions, the probability (4.3) becomes

$$Pr[L_2 = l' | L_1 = l] \tag{4.5}$$

for $l, l' \in [1, h]$. Therefore, the problem of predicting the future consists in finding these conditional probabilities and their distributions.

4.2.1 Estimation of Conditional Probability Density

This paragraph describes an *off-line* estimation of the conditional probabilities introduced in the previous paragraph.

Given a data traffic trace X_t collected in a Ethernet LAN, the first step consists in segmenting this series into

$$N = \frac{n}{T_1 + T_2}$$

contiguous non overlapping blocks of length $T_1 + T_2$ where n is the number of samples of the trace (time domain is discrete).

The second step is the computation of the aggregate traffic V_1, V_2 for each $j \in [1, N]$ over the sub-intervals of length T_1, T_2.

After the choice of the number of levels h for the quantization of the traffic, the third step consists in counting the number of blocks $h_l \in [0, N]$ such that $L_1(V_1) = l$ and then the number of those blocks $h_{l'} \in [0, h_l]$ such that $L_2(V_2) = l'$. Using this method the conditional probability can be calculated easily as:

$$Pr[L_2 = l' | L_1 = l] = \frac{h_{l'}}{h_l}. \tag{4.6}$$

This method is proposed in [**14**], Cap.18, where also some interesting experimental results are reported. In [**14**] two cases are studied: $\alpha = 1.05$ which means $H = 0.975$ (very bursty traffic) and $\alpha = 1.95$ which means $H = 0.525$ (less bursty traffic). In the first case the conditional probability densities have this property: if the traffic L_1 is low, most likely also L_2 will be low; conversely, if the traffic L_1 is high, most likely also L_2 will be high. In the second case, instead, the shape of the distributions does not change when the conditioning variable L_1 is varied. This indicates that observing the past does not help in predicting the future.

These experimental results show what has been already explained: a traffic with a high degree of self-similarity can be predicted with reliability otherwise only some a priori information (if there is some) can be exploited. Fig. 4.1 shows the conditional probabilities studied in [**14**] in the two different cases explained.

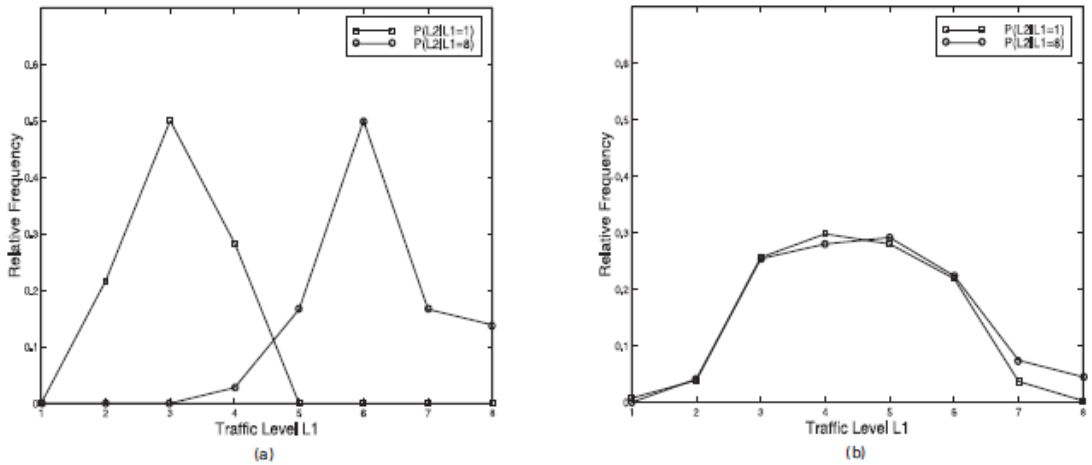

Figure 4.1: *Conditional probabilities $Pr[L_2|L_1 = l]$ and $Pr[L_2|L_1 = 8]$ in the two cases: (a) corresponds to $H = 0.975$,(b) corresponds to $H = 0.525$.*

4.2.2 Predictability and Time Scale

There is another important topic to consider, that is, how time scale affects predictability. In fact another study, always presented in [**14**], Cap. 18, shows that as time scale is increased the conditional probability densities $Pr[L_2|L_1 = l]$ become more concentrated. In this paragraph the question about at what time scale predictability is maximized is answered.

To solve the problem a different mathematical tool is introduced, that is, *entropy*.

In information theory (and in relation to the signal theory), entropy measures the amount of uncertainty or information found in a random signal. From another point of view, the entropy is the minimum descriptive complexity of a random variable, which is the lower limit of the data compression without loss of information. The connection with the thermodynamic entropy is in the compression ratio: the decrease of temperature corresponds to the reduction of the redundancy of the signal, and therefore the increase of the compression. The information entropy reaches a minimum which, in general, is different from zero. For a discrete probability density p_i, its entropy $S(p_i)$ is defined as $S(p_i) = -\sum_i p_i log(p_i)$. In this case of the conditional density, the entropy is defined as

$$S_l = -\sum_{l'=1}^{h} \Big(Pr[L_2|L_1 = l] \Big) \cdot log(Pr[L_2|L_1 = l]). \tag{4.7}$$

and since there are h different entropies (one for each traffic level), also the *average entropy* $\bar{S}$ is defined:

$$\bar{S} = \sum_{l=1}^{h} S_l/h. \tag{4.8}$$

However it can be observed that when the distribution of probability is uniform, the entropy is maximal while when the distribution of probability is concentrated in a single point, the entropy is minimal. Here another important result is presented: from the studies in [**14**], average entropy is highest for small time scales and it drops as T_1 is increased. Naturally T_1 could be increased further and further to gain small decreases in entropy but the information may not be effectively exploitable if the time interval is too long. Therefore, choosing T_1 small, the prediction about future traffic will not be so precise (high entropy) while for a high T_1, although the prediction is more precise, time interval is too long for exploitable information.

4.2.3 On-Line Estimation of Future Traffic

In the previous paragraphs, the analysis are done following an off-line approach while in reality it would be interesting to predict future traffic while traffic itself is still arriving. For this reason there is the need of an on-line prediction. This problem can be solved using $O(1)$ cost update operations, with an on-line estimation of the conditional probability density. This can be done building a table which can be called *CondProb* of size h x $h+1$, one row for each $l \in [1, h]$. The last column of *CondProb* [l][h+1] is used to keep track of h_l, the number of blocks observed so far whose traffic level maps to l. For each $l' \in [1, h]$, *CondProb*[l][l'] maintains the count

$h_{l'}$. Therefore having *CondProb* means having the conditional probability densities. Then it is necessary a clock starting at time $t = 0$ and which goes off at times

$$t = T_1, T_1 + T_2, T_1 + T_2 + T_1, T_1 + T_2 + T_1 + T_2...$$

Therefore all the packets arriving during the periods

$$[i(T_1 + T_2), i(T_1 + T_2) + T_1], \quad i \geq 0$$

are added to V_1. When the clock goes off at $t = i(T_1 + T_2) + T_1$, V_1 is used to compute the updated V^t_{max} and V^t_{min} and the quantization step μ. Now $l = L_1(V_1)$ is computed using the updated V^t_{max} and V^t_{min} and *CondProb*[l][h+1] is incremented by 1. During interval

$$[i(T_1 + T_2) + T_1, (i + 1)(T_1 + T_2)], \quad i \geq 0$$

a similar operation is done respect to V_2. At the end of the interval, the updated V^t_{max} and V^t_{min} are computed and also $l' = L_2(V_2)$ is computed. Finally *CondProb*[l][l'] is incremented by 1 and V_1, V_2 are reset to 0 to start the process again.

Clearly, these conditional densities computed from *CondProb* are approximations of the corresponding ones computed with the off-line method since in the on-online algorithm running sums are used to compute and update V^t_{max} and V^t_{min}.

Convergence of On-Line Conditional Probabilities

Before using the table *CondProb* of the conditional probabilities, it is necessary that the estimations on-line computed converge or stabilize otherwise the prediction of the future traffic could not be accurate and instead could hurt the performance of the algorithm. For this reason, it is useful to introduce a distance measure to decide whether a particular conditional probability is stable enough to be used for a correct prediction of the future. Let $CondProb_{t_1}$ and $CondProb_{t_2}$ be two instances of the conditional probability density table measured at time instances $t_2 > t_1$ at least $T_1 + T_2$ away. Then for each $l \in [1, h]$, there must be verified that

$$||CondProb_{t_2}[l][\cdot] - CondProb_{t_1}[l][\cdot]||_2 < \Theta. \tag{4.9}$$

where $\Theta > 0$ is an accuracy parameter which rules the precision of the distance between the two vectors of probabilities. If the inequality is satisfied, the l-th conditional probability is stabilized and can be used for the prediction. In Fig.4.2 it is shown an example of the trends of the distances computed for two conditional probabilities.

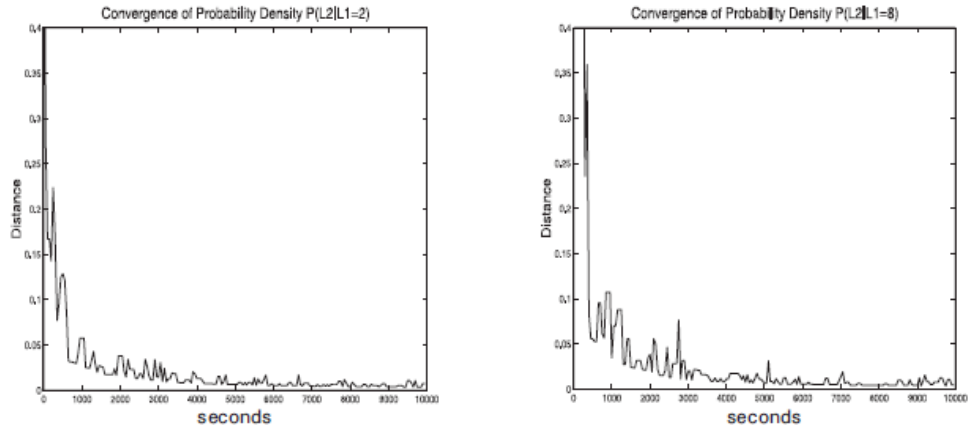

Figure 4.2: *Examples of two different conditional probabilities (taken from [**14**])*

4.3 SAC and Underlying Congestion Control

In this section a congestion control strategy called selective aggressiveness control (SAC) is presented (all the details in [**14**]). This technique is particularly interesting because it exploits the predictability structure present in long-range dependent traffic. In fact a certain control action $\epsilon(l)$ is taken using the information about the future traffic and this action will affect the decisions of an underlying congestion control.

SAC has the aim to improve the performance of existing congestion controls. Setting up a simple rate-based feedback congestion control as a reference, SAC always respects the taken decision by this simple control (increase or decrease of traffic rate) but it can modify the magnitude of this change. For example, if the underlying control decides to increase the traffic rate, SAC can decide of how much increase it.

4.3.1 Underlying Congestion Control

For simplicity, a generic instance of rate-based feedback congestion control will be employed as a reference to demonstrate the efficacy of selective aggressiveness control under self-similar conditions.

Let λ denote packet arrival rate and let γ denote throughput. The generic feedback congestion control has a control law of the form:

$$\frac{d\lambda}{dt} = \begin{cases} \delta, & \text{if } \frac{d\gamma}{d\lambda} > 0 \\ -a\lambda, & \text{if } \frac{d\gamma}{d\lambda} < 0 \end{cases} \tag{4.10}$$

where δ, a are positive constants. Thus, if increasing the data rate, also the throughput raises, then this control increases the data rate linearly; conversely, if increasing the data rate, the throughput reduces, then this control decreases the data rate exponentially.

Before continuing, it is necessary to better explain what the throughput γ is. Throughput can be defined in different ways depending on the context: there is the reliable throughput (number of bits reliably transferred per unit time) or the raw throughput (number of bits transferred per unit time) or the power (one of the throughput measures divided by delay). Raw throughput, denoted ν, is easy to measure (just monitor the number of packets, in bytes, arriving at the receiver per unit time) but it is not adequate for this kind of measures (see [**14**] for the motivations). Instead the measure of throughput that will be used is

$$\gamma_k = (1 - c)^k \nu$$

that penalizes raw throughput ν by packet loss rate $0 \leq c \leq 1$ where the severity can be set by the parameter $k \geq 0$. Therefore, the throughput that appears in (4.10) is γ_k, measured at the receiver.

4.3.2 Selective Aggressiveness Control (SAC)

Assuming that future network traffic is predictable with a sufficient degree of accuracy, there remains the question of what to do with this information. The choice of actions depends on the networking context and what degree of freedom it allows. Usually the only control variable available is its traffic rate λ.

Given the linear increase/exponential decrease control in (4.10), SAC tries to work on the linear increase part such that a more aggressive bandwidth consumption is facilitated. This selective application of aggressiveness, when coupled with predictive capability, will hopefully lead to a more effective use of bandwidth with improved performance.

SAC is composed of two parts, prediction (already described in the previous paragraph "On-Line Estimation of Future Traffic") and application of aggression that is now described.

Let λ_t denote the newly updated rate value at time t and let $\lambda_{t'}$ be the most recently ($t' < t$) updated value previous to t. SAC behaves in this way:

- If $\lambda_t > \lambda_{t'}$ then update $\lambda_t \leftarrow \lambda_t + \epsilon_t$.
- Else do nothing.

Here $\epsilon_t \geq 0$ is an aggressiveness factor that is determined using the current state of the table of conditional probabilities *CondProb*. It is worth to note that SAC kicks into action only during the linear increase phase. The magnitude of ϵ_t determines the degree of aggressiveness and it is determined as a function of the predicted network state as captured by *CondProb*.

At time t, ϵ_t is determined in this way.

1. Let S_t be the aggregate throughput reported by the receiver via feedback over time interval $[t - T_1, t]$.
2. Let $l = L_1(S_t)$.
3. Compute $\bar{l}' = E(L_2|L_1 = l) = \sum_{l'=1}^{h} l' \cdot Pr[L_2 = l'|L_1 = l]$.
4. Set $\epsilon_t = \epsilon(\bar{l}')$.

Thus, the current traffic level S_t is normalized and mapped to the index l which is used to calculate the expectation of L_2 conditioned on l'. This expectation is then used to compute $\epsilon(\bar{l}')$, called aggressiveness schedule. The idea is that if the expected future traffic is low, then it is preferable to use a high level of aggressiveness; conversely, if the expected future traffic is high, then it is preferable to use a low level of aggressiveness. One schedule that can be used is the inverse schedule:

$$\epsilon(\bar{l}') = 1/\bar{l}'. \tag{4.11}$$

Other schedules of interest include the threshold schedule with threshold $\theta \in [1, h]$ and aggressiveness factor θ^*, where $\epsilon = \theta^*$ if $\bar{l}' \leq \theta$ and 0 otherwise (for interesting simulations results see [**14**]).

SAC essentially works by varying the arrival rate but it is not always possible to change this quantity. This is the case that will be presented in the chapter dedicated to the simulations and for this reason a new strategy is now presented.

4.4 EEEP Strategy

The new developed algorithm merges and puts its bases on the main two topics discussed in this thesis, that is, the self-similarity of a LAN traffic, in particular, the property of long-range dependence with the possibility to make a prediction of the future traffic and the recent researches about Energy Efficiency with the development of the IEEE 802.3az standard and LPI mode.

This paragraph represents the heart of the thesis because it explains all the steps with all the details of the working of the algorithm, which is called EEEP strategy, that is, Energy Efficient Ethernet Prediction strategy. In Fig.4.3 it is reported the scheme of the algorithm. As first step, some parameters have to be fixed: T, the period for using EEE technique, T_1, which, as already explained, represents the first part of the blocks used for the prediction strategy (for simplicity $T_1 = T_2$, therefore T_1 represents half of every block), Θ, used as accuracy parameter (see previous paragraph), LW, which determines the length of the window for the computation of the Hurst parameter, ND, which represents how many new data in LW have to be chosen for the computation of H.
When data traffic starts, the algorithm adopts the EEE strategy described in Chapter 3 on the period T and starts to build the On-Line table of the conditional probabilities with updates every T_1 seconds. The second step consists in calculating the distances between corresponding rows of the table at two different instances, as described in the previous paragraph, and, if these distances are lower than the parameter Θ, the algorithm goes on with the third step in which there is the computation of the Hurst parameter. If $H > 0.6$ (the value 0.5 is too low to apply the strategy, therefore it is better to consider greater values), the prediction strategy can be applied. This technique works in this way: considering the data traffic divided in blocks of length $2T_1$, in the first part of each block normal EEE strategy is applied. Then, if the expected traffic in the second part of the block is lower or equal to that one measured in the first part, the link, which has to transmit data, turns off when the first part ends. The assumption that can be made is that the time used for the transmission of the packets in the first part of the block is similar or lower (as value) to that one that will be necessary to transmit the packets in the second part of the block. Therefore, the link can be turned on just the calculated time before of the end of second part of the block; in this part, the link will transmit (with the link always on, not using EEE policy) all the packets that in the meantime when the link was off have accumulated in the queue of the buffer. Clearly, when the link turns on after

the time that it was off, other packets arrive in this last part of the block and the algorithm, as it is implemented, decides to immediately transmit these packets in the first part of the T period and, in the remaining part of the period, the largest amount of packets in the queue are transmitted until the next period T.

Using this policy, there are now two situations that the algorithm can meet.

1. The queue of the packets is depleted before the end of the block: in this case normal transmissions happen and then, when the new block starts, the algorithm begins all the steps described again.

2. The packets in the queue cannot be all transmitted within the current block because the time calculated is not sufficient. To avoid packets loss, the algorithm decides to transmit these remaining packets of the queue in the first part of the new block which restarts the use of EEE policy.

Anyway, all this strategy is not exploited if the predicted future traffic in the second part of the block is greater than the the traffic in the first part and an EEE policy is adopted. The prediction strategy can be applied if the traffic actually is self-similar, that is, if the Hurst parameter is greater than 0.6. For this reason, there is the need to control that H is effectively greater than 0.6. This is what the algorithm does as last step. If there are new data which have to be transmitted, the algorithm computes H using windows of LW data including $LW - ND$ old data and ND new data. If the computation of H gives a value greater than 0.6 the prediction strategy can be again applied otherwise only an EEE policy will be used.

The great benefit of the prediction strategy consists in limiting the interval of time in which the link is on. As in the chapter dedicated to the simulations it will be discussed, this technique permits to avoid all the transitions which occur in the EEE policy with great energy savings. In fact it was demonstrated that the power lost in the transitions is nearly the same of that one consumed in the active state. On the other side, there is also a drawback which characterizes this algorithm: in fact the packets, which are not immediately transmitted because the link is off and which are accumulated in the queue, will be transmitted with some delay which can be important if the window T_1 is large. However for applications that have not strict time requirements, this technique can be very useful and energy efficient (with a maximum delay of packets transmission equal to T_1).
In Fig.4.3 the main steps explained in this paragraph are illustrated in the flow chart.

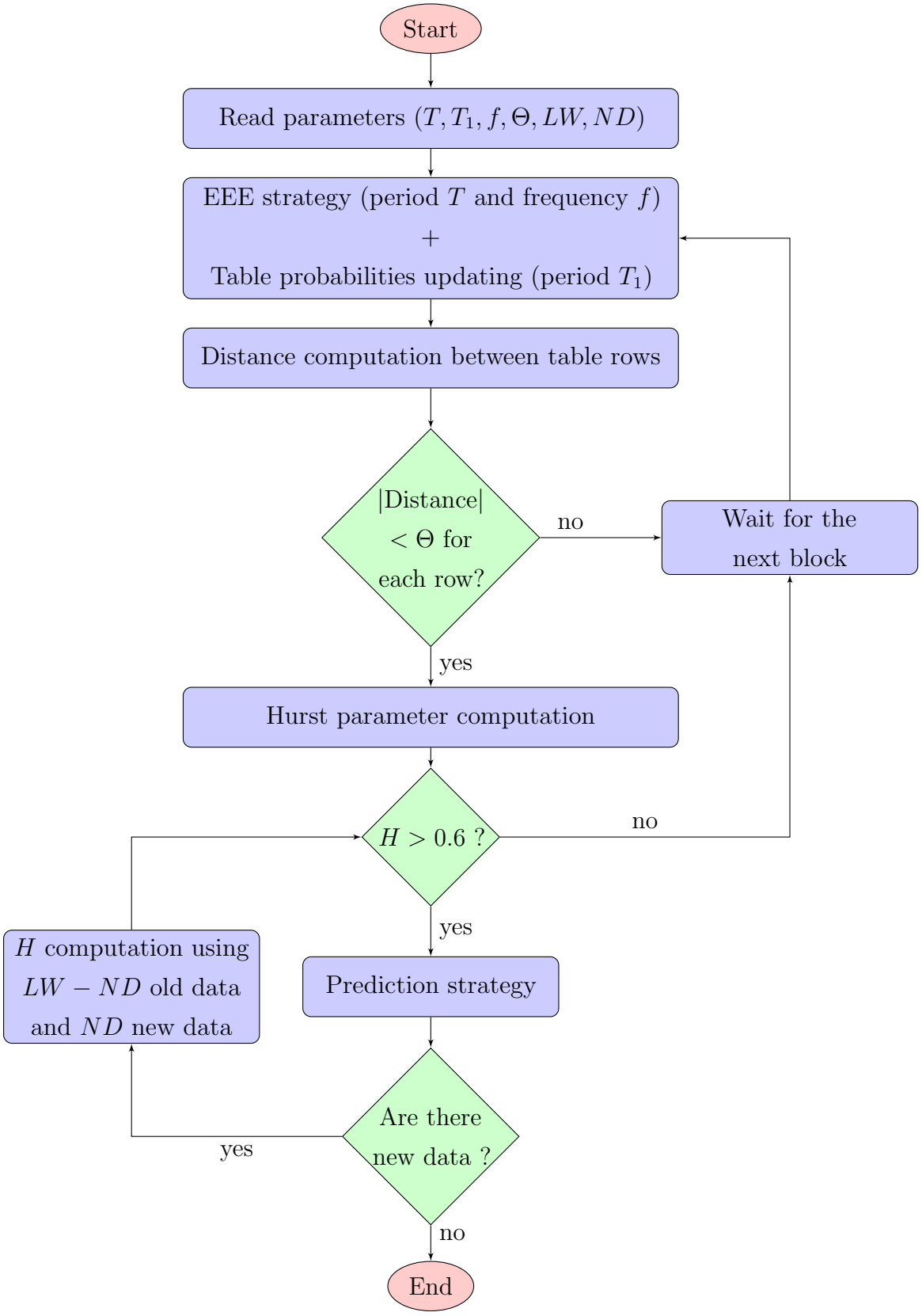

Figure 4.3: *Scheme of the algorithm.*

In Fig.4.4 and in Fig.4.5 there are two schematic representations of the working of the algorithm.

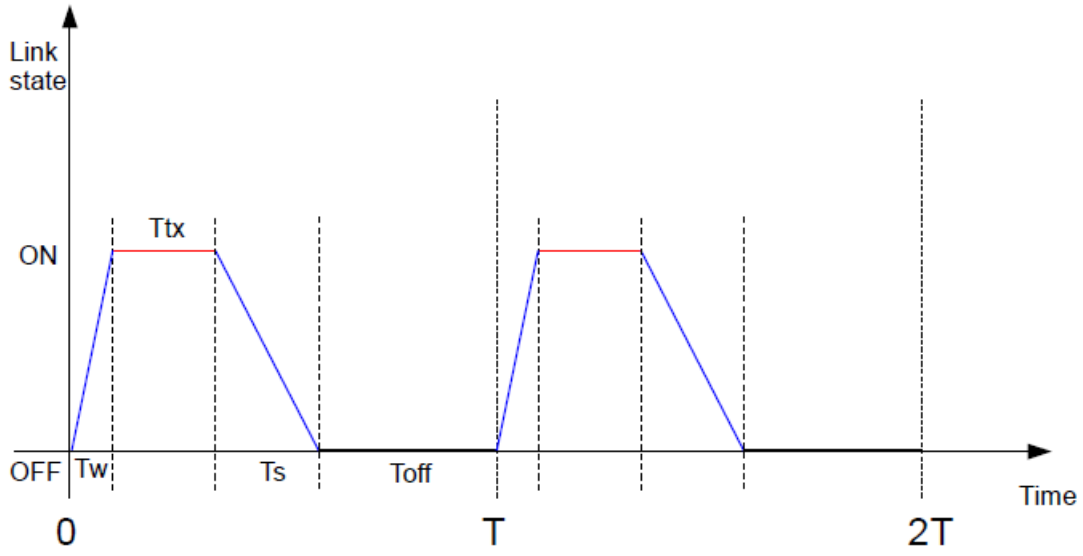

Figure 4.4: *Working of the algorithm on 2 periods of the first part of the block (EEE strategy).*

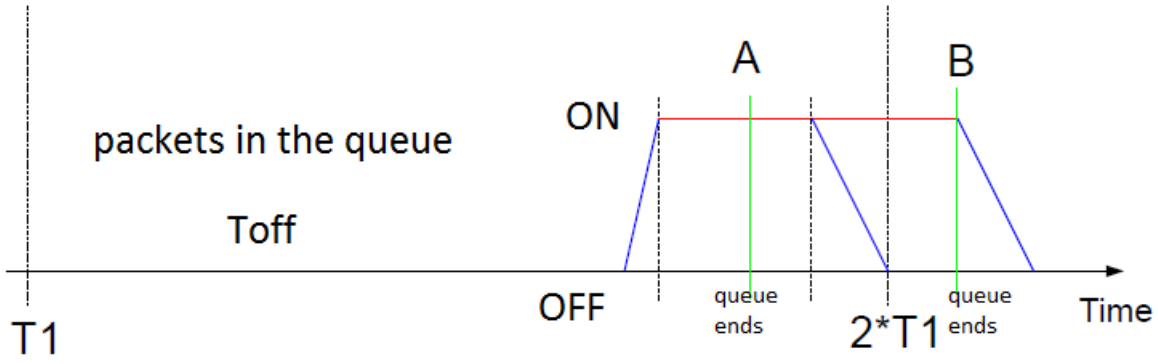

Figure 4.5: *Working of the algorithm on second part of the block.*

In Fig.4.4 there is the representation of the behavior of the algorithm on 2 periods of the first part of a block: every T sec, the link turns on in t_w sec, transmits in t_{tx} sec and turns off in t_s sec and then it stays off until the beginning of the new period.

In Fig.4.5 there is the representation of the behavior of the algorithm in the second part of a block which uses prediction strategy. The link stays off for great part of the period T_1, then turns on, transmits all the packets accumulated in the queue and

here two situations are depicted: the case A in which the queue ends before of the block and case B in which the queue ends in the subsequent block.

The case in which a block does not use prediction is not illustrated because the behavior is the same of that one reported in Fig.4.4 for all the length of the block.

4.5 Expected Results

Before the beginning of the simulations, it can be useful to find some relations among the used parameters to have an idea of what it can be expected.

Here, for correctness, also the inter-frame gap (IFG) will be considered. Briefly, Ethernet devices have to allow a minimum idle period period between transmission of Ethernet frames and this period is known as inter-frame gap: its value is equal to 96 bit which, with the frequency chosen (1 Gbit/s), can be transmitted in 0.096 μs. Since this time is so small, for reason of simplicity, in the algorithm it will not be considered. It will be considered here only for correct calculations. What, instead, is very meaningful to consider, is the time which EEE strategy spends for the transitions (t_w and t_s) which is equal to $t_w = 0.0165\,ms$ and $t_s = 0.202\,ms$ respectively. Finally, also the packet size d is important because, clearly, if the packet size is low, more packets can be transmitted in the same interval. An Ethernet frame can have different dimensions depending on how many bits it brings and the minimum frame size is fixed to $m = 512\,bit$ while the maximum size is equal to $M = 12000\,bit$. Let N be the number of consecutive packets which can be transmitted in the interval T.

If the link is always on, this equation can be written:

$$N = \left\lfloor \frac{f \cdot T}{d + IFG} \right\rfloor \tag{4.12}$$

If the link adopts EEE strategy, these two relations can be written:

$$N = \left\lfloor \frac{f \cdot (T - T_{trans} - perc \cdot T/100)}{d + IFG} \right\rfloor \tag{4.13}$$

$$perc = 100 \cdot \frac{(T - T_{trans} - \frac{N(d+IFG)}{f})}{T} \tag{4.14}$$

where $T_{trans} = t_w + t_s$ and $perc$ represents the percentage of the time T in which the link is off. Since the average value of packets transmitted in T is $\bar{N} = 13$ and the average value of the packets size with IFG is $\bar{d} + IFG = 5780\,bit$, using the last relation, the percentage of time in which the link will be off is expected to be about $perc = 70.6\%$.

Chapter 5

Simulations Results

This chapter is dedicated to the simulations realized with the software MATLAB in order to show the results obtained implementing the algorithm explained in the previous chapter. First of all, it is necessary to work on traffic traces which show self-similar characteristics. There are two ways to obtain this kind of traces: the first way is measuring data traffic on a real network while the second way is generating traffic in artificial mode. In this thesis the first way is considered but it can be very useful to consider also the artificial generation of network traffic.

5.1 Artificial Generation of the Network Traffic

This paragraph briefly describes how to generate artificial network traffic, well studied in [**7**]. An artificial traffic generator with parameters which can be modified to obtain different self-similarity degrees, can be built using chaotic systems. To see some general properties of this kind of systems and how these systems are related to the self-similar conditions, see [**7**].

Here a map to generate the samples of a self-similar process is presented. This map is defined as $f : [0, 1] \to [0, 1]$ and it is built splitting a piece-wise Markov map [1] into an infinite set of intervals $[z_i, z_{i-1}[$, with $z_0 = 1$, $z_i < z_{i-1}$ and $z_i \to 0$ for $i \to \infty$ as:

[1] f is a piece-wise affine Markov map, with respect to the quantization intervals, if f, restricted to any interval X_j, is affine and if, for any X_{j1} and X_{j2}, either $f(X_{j1}) \cap X_{j2} = \emptyset$ or $X_{j2} \subseteq f(X_{j1})$.

$$f(x) = \begin{cases} \frac{x-z_1}{1-z_1}, & x \in [z_1, 1[\\ z_{i-1} + (x - z_i)\frac{z_{i-2}-z_{i-1}}{z_{i-1}-z_i}, & x \in [z_i, z_{i-1}[, \quad i \geq 2 \end{cases} \tag{5.1}$$

where $z_i = (i+1)^{2H-3}$ with H, the Hurst parameter which can be properly varied. Then, the final artificial traffic source can be obtained using a non-linear quantization function $Q : [0, 1] \to \{0, 1\}$ with $Q(y) = 0$ if $y \in [0, z_1[$ and $Q(y) = 1$ if $y \in [z_1, 1[$: in this way the source will be off in the interval $[z_\infty, z_1[$ and it will be on in the interval $[z_1, z_0[$. To explore the topic in the detail see [**7**].

5.2 Real Traffic Data

This paragraph presents some analysis made on two kinds of traffic series. These traffic traces were collected at some of the most important USA laboratories, measuring data traffic in some links of the networks which researchers had available.

5.2.1 AuckVIII Series

This first kind of series was used to test one of the inference technique presented, that is, the variances analysis in order to find the Hurst parameter. Briefly, in this set called AuckVIII, there are 24 traffic traces each with 72000 data in which packets are collected with a time granularity of 50 ms. These 24 traces represent the traffic measured every hour of the day. In Fig.5.1 is reported the trend of the Hurst parameter as function of the hours of the day.

As it can be seen from the figure 5.1, traffic during the day maintains its self-similar behavior with an average Hurst parameter equal to 0.83. It is interesting to note the peak at 7 am: maybe the behavior of the network is very bursty. In fact, with the beginning of the working day, the network is supposedly more used than the night hours in which the average traffic is low with only some peaks sometimes.

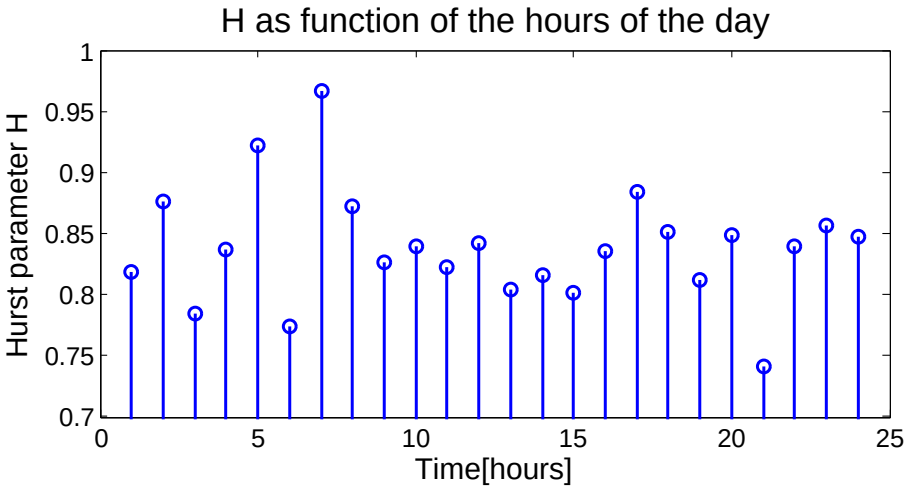

Figure 5.1: *Hurst parameter as function of the hours of the day.*

5.2.2 San Diego Series

This second kind of series was used to test the algorithm presented above. In this set, called Sandiego, there are 6 traces each with 300000 data collected every millisecond. As the previous set, files contains the number of packets with the corresponding bytes received by a host in the network.

In this thesis the analyses are mainly made on a single trace. In Fig.5.2 it is reported the complete trace considered with all packets received and the corresponding variance analysis to find the Hurst parameter.

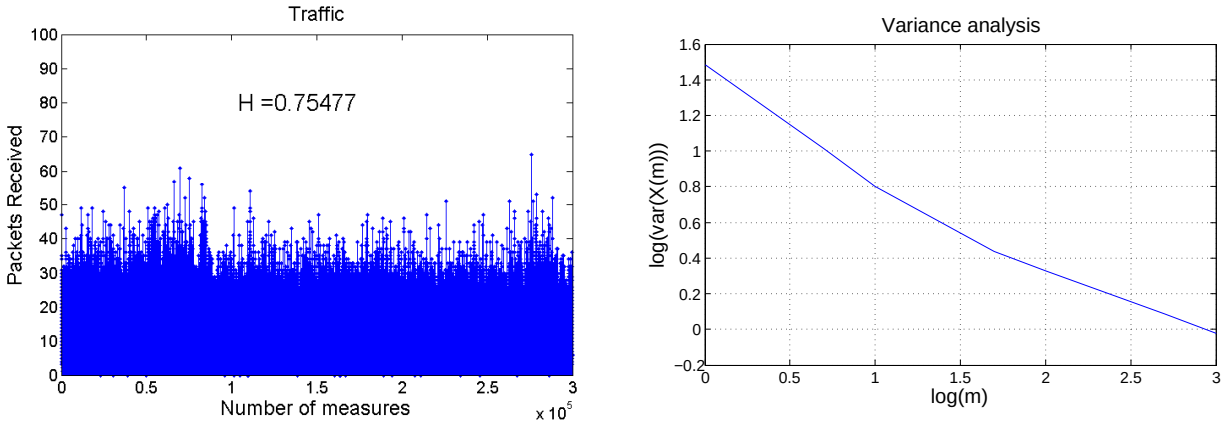

Figure 5.2: *Complete trace (left) with variance analysis (right).*

Since all the traffic was known, simulations do not create the building of the table of conditional probabilities On-Line but they built it before the start of the

algorithm. With this particular trace, the choice of the number of quantization levels to classify the traffic fell on 8.
Here it is reported the table of the conditional probabilities and the corresponding vector of average values which is used by the algorithm to decide if applying the prediction strategy or not. Position (i, j) of the table indicates the probability that $L_2 = j$ given that $L_1 = i$. In the last column there are the average values for each row: this indicates that, given the traffic level l of the first part of the block, on average, the level of traffic l' is expected to appear in the second part.

	1	2	3	4	5	6	7	8	$E[L_2\|\cdot]$
1	0.12	0.52	0.32	0.04	0	0	0	0	2.3
2	0.0372	0.3716	0.4730	0.0980	0.0135	0.0068	0	0	2.7
3	0.0189	0.2346	0.4756	0.2331	0.0283	0.0063	0.0031	0	3.0
4	0.0054	0.0790	0.4251	0.3678	0.1035	0.0191	0	0	3.5
5	0	0.0504	0.2941	0.3866	0.1849	0.0756	0.0084	0	4.0
6	0	0	0.1143	0.1714	0.2	0.2286	0.2286	0.0571	5.5
7	0	0	0.0526	0.0526	0.2632	0.2632	0.2632	0.1053	6.0
8	0	0	0	0	0	0.5	0.5	0	6.5

Table 5.1: *Table of conditional probabilities.*

In Fig.5.3 the trend of the eight conditional probabilities is reported.

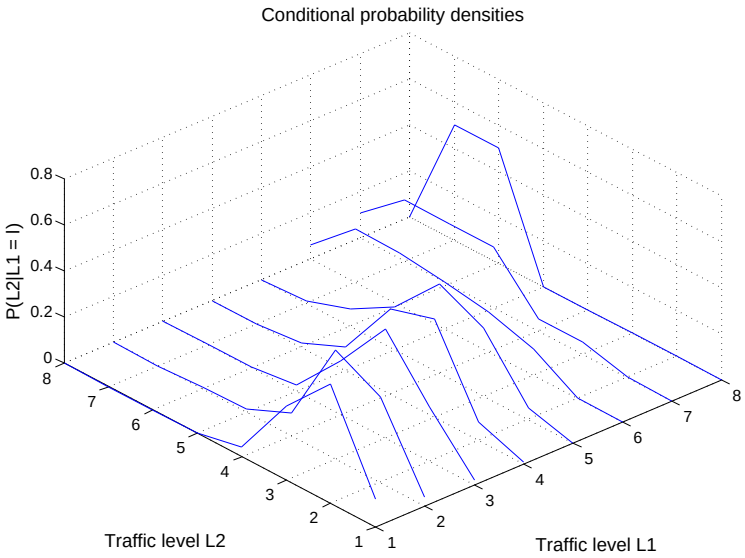

Figure 5.3: *Conditional probability densities.*

5.3 Parameter tuning for EEEP

Before the beginning of the algorithm, it is necessary to choose some parameters as the flow chart in the previous chapter well explains.

T is already fixed because measures are collected with a period of 1 ms: therefore EEE strategy will be applied every millisecond. Also the frequency of the data flow is fixed and it is equal to $1Gbit/s$.

The parameters which have to be chosen are:

- T_1, that is, the length of the window for the prediction strategy,
- LW, that is, the number of data considered for Hurst parameter computation,
- ND, that is, how many new data have to be considered in LW for the computation of H.

The choice of T_1 can be made considering the analysis previously done on two other traces. The number of data considered for both traces is 200000. The EEEP strategy was simulated choosing different values of T_1 and, for each T_1, different increments of time respect the useful time calculated for the packets transmission are considered. This is motivated by the fact that fewer the samples used for prediction, the less accurate this prediction is and hence the need of increase the calculated time. In Tab.5.2 and Tab.5.3 there are some useful data collected from simulations which can give some indications for the choice of T_1.

These tables have 5 columns: in the first one there are various values of T_1 chosen. In the second one there are the percentages of increment of the useful time calculated for the transmission of packets in the second part of the blocks when prediction strategy is adopted. In the third one there are the percentages of blocks for which the useful time calculated (with its increment) is not sufficient to transmit all the packets. Finally, in the fourth and in the fifth ones are reported, respectively, the time gains and the energy gains (in percentage) respect to the only EEE strategy. In Tab. 5.2 and in Tab. 5.3 there are reported the results of one trace using eight values for T_1.

Windows Sizes T_1 (ms)	Increments of useful time (%)	Blocks with useful time not sufficient (%)	Time gains (%)	Energy gains (%)
50 ms	+ 0	42.74	29.92	23.41
(2000 blocks)	+ 10	22.22	28.77	22.5
82 % of blocks	+ 20	9.4	27.72	21.68
use prediction	+ 30	3.85	26.52	20.75
	+ 40	1.89	25.42	19.89
	+ 50	0.85	24.31	19.02
	+ 60	0.24	23.19	18.14
	+ 70	0.06	22.04	17.24
	+ 80	0	20.93	16.37
100 ms	+ 0	43.26	29.82	23.32
(1000 blocks)	+ 10	18.26	28.72	22.46
82 % of blocks	+ 20	5.64	27.64	21.63
use prediction	+ 30	1.84	26.5	20.74
	+ 40	0.49	25.37	19.89
	+ 50	0.12	24.27	18.98
	+ 60	0	23.14	18.11

Table 5.2: *Windows from 50ms to 100ms*

Some considerations can be done observing the tables: it can be said that, in general, increasing the useful time for the transmission of packets of certain percentages, the number of blocks which are not able to transmit packets in the useful time computed decreases with the corresponding decreases of time and energy gains. Another consideration is that, for short windows, in order to correct transmit all the packets of all the blocks, it is necessary to increase the predicted useful time of high percentages while for long windows it is sufficient only, for example in the case of 5000 ms, a 20 %. It seems that long windows are safer regarding the time predicted for the transmission (only few increments of time to obtain correct transmission) but they have a great drawback, that is, they introduce great delay to the packets waiting in the queue. Finally, the lowest time and energy gains belong to the window of 2500 ms: probably this behavior is due to the fact that less of 60 % of blocks use prediction strategy and therefore the benefits which this strategy brings are not exploited. These results are referred to the first trace analyzed while those ones of the other trace, since they are similar, are reported in Appendix A. Therefore, after

Windows Sizes T_1 (ms)	Increments of useful time (%)	Blocks with useful time not sufficient (%)	Time gains (%)	Energy gains (%)
200 ms	+ 0	46.56	28.83	22.55
(500 blocks)	+ 10	15.78	27.76	21.71
79 % of blocks	+ 20	4.83	26.67	20.87
use prediction	+ 30	1.02	25.6	20.04
	+ 40	0.25	24.53	19.19
	+ 50	0	23.46	18.35
500 ms	+ 0	37.32	25.98	20.32
(200 blocks)	+ 10	11.27	25	19.56
71 % of blocks	+ 20	3.52	24.04	18.8
use prediction	+ 30	0.7	23.07	18.06
	+ 40	0	22.09	17.28
1000 ms	+ 0	37.5	26.28	20.56
(100 blocks)	+ 10	9.72	25.3	19.79
72 % of blocks	+ 20	1.39	24.31	19.02
use prediction	+ 30	0	23.33	18.25
2000 ms	+ 0	48.89	33.28	26.04
(50 blocks)	+ 10	15.56	32.1	25.11
90 % of blocks	+ 20	2.22	30.92	24.19
use prediction	+ 30	0	29.73	23.26
2500 ms	+ 0	56.52	21.23	16.61
(40 blocks)	+ 10	21.74	20.47	16.02
58 % of blocks	+ 20	0	19.73	15.43
use prediction				
5000 ms	+ 0	38.89	32.93	25.76
(20 blocks)	+ 10	5.56	31.73	24.82
90 % of blocks	+ 20	0	30.53	23.88
use prediction				

Table 5.3: *Windows from 200ms to 5000ms*

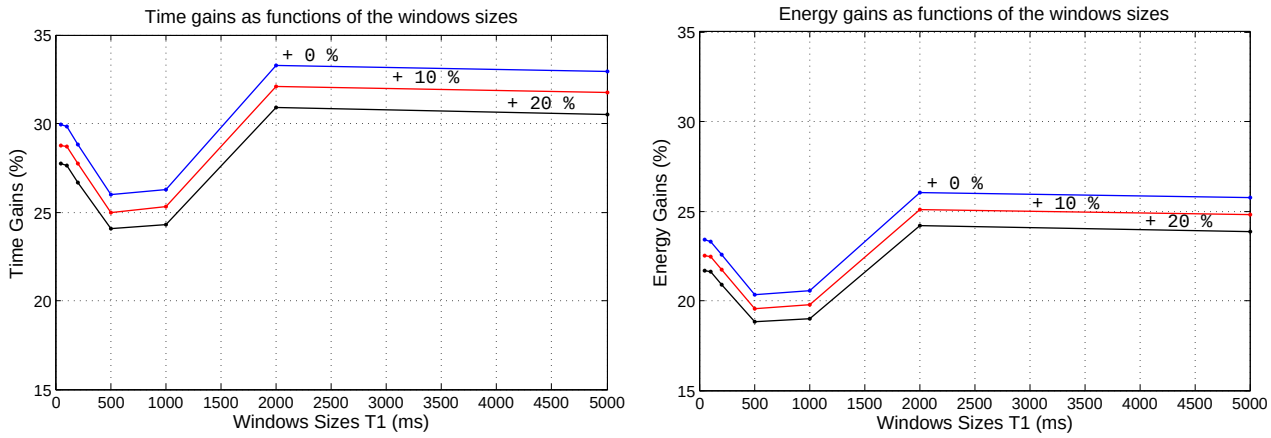

Figure 5.4: *Time and energy gains (%) using different windows sizes. Blue, red, black lines correspond respectively to +0 %, +10 %, +20 % of increment of useful time.*

these observations, the choice of T_1 fell on $T_1 = 100ms$ in order not to bring much delay to the packets.

Another decision to make is the choice of LW and ND. In fact there is the need to control, computing the Hurst parameter, that the arriving traffic data maintain the self-similarity properties. As done for the choice of T_1, it can be useful to study the effects that different choices of LW and ND have in other traffic traces. In the following figures, six values of LW are tested: $LW = 100000, 50000, 20000, 10000, 5000, 2000$ and, moreover, for each LW chosen, various values of ND are tried. Each of the following figures represents the trend of H, computed on LW data of which ND are new and the overlapped graphics of different colors (overlapped because it is better for a comparison) are the trends of H considering different choices for ND.

Some considerations can be drawn: for high values of LW, H has a quite regular trend and it fluctuates in a short range of values while for short windows, H varies a lot with an abrupt trend. Another observation is that, once chosen LW, H behaves in the same way regardless of how many new data are considered (although trend is more sensible to few new data). Therefore, using a high values for LW, the probability that the trend of H presents similar values during the simulations is high while it is the opposite for short windows and, moreover, the time used for the computation of H is high for high windows, the opposite for short ones. To achieve a good compromise, the choice fell on $LW = 20000$ and $ND = 1000$ in order not to have so much data (time for the computation of H low) and to control the traffic quite frequently.

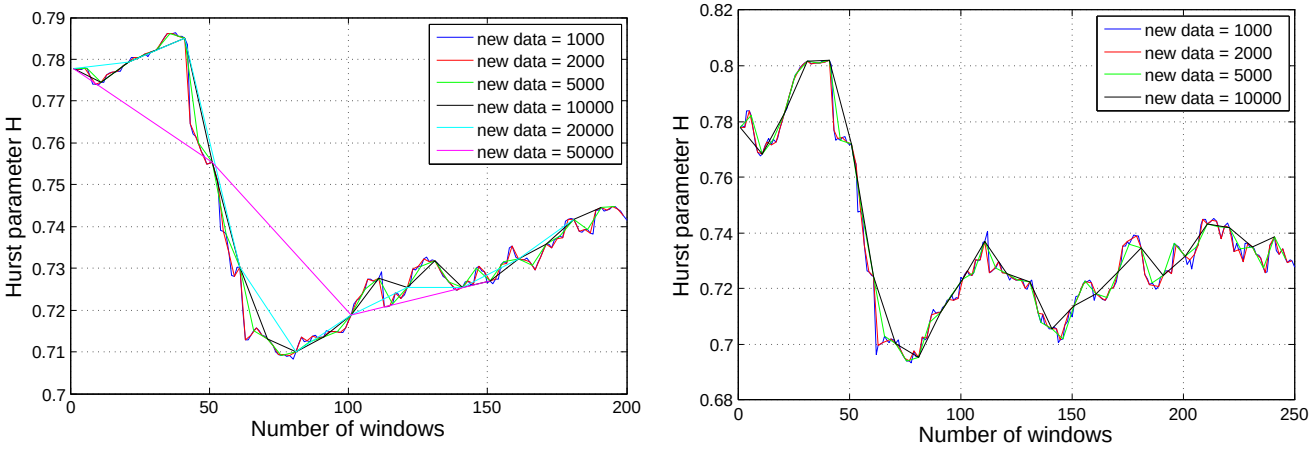

Figure 5.5: *Window length = 100000 data (on the left) and 50000 data (on the right)*

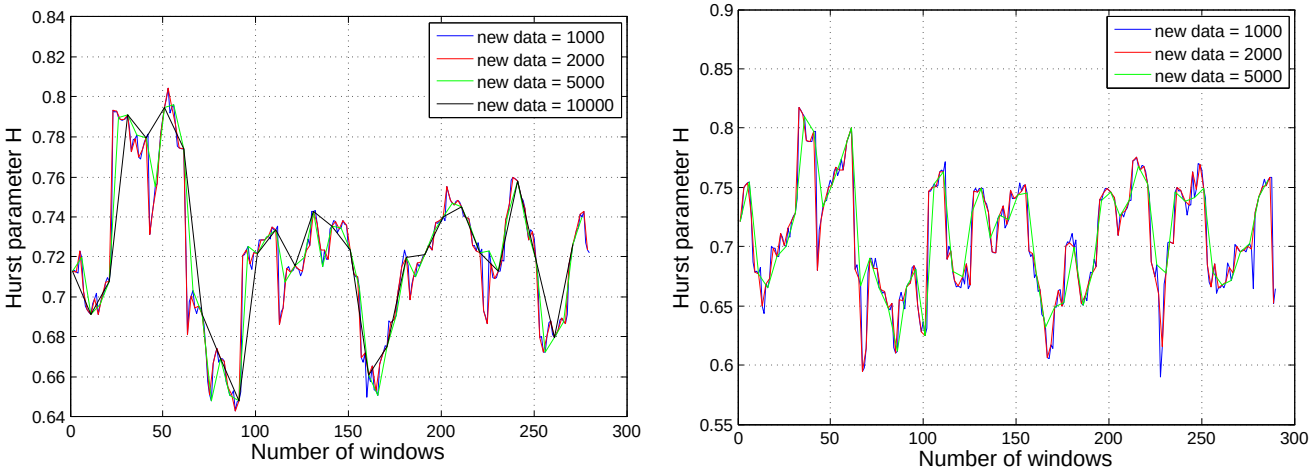

Figure 5.6: *Window length = 20000 data (on the left) and 10000 data (on the right)*

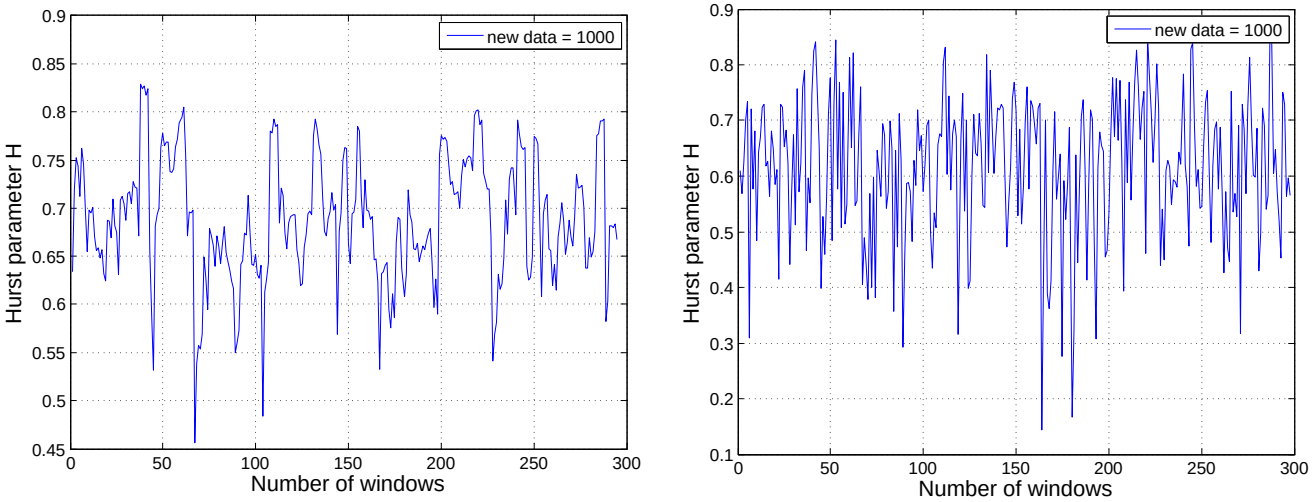

Figure 5.7: *Window length = 5000 data (on the left) and 2000 data (on the right)*

To sum up the found results, in the simulations these parameters will be used:

$$T = 1\,ms$$
$$f = 10^6\,bit/ms$$
$$T_1 = 100\,ms$$
$$LW = 20000\,data$$
$$ND = 1000\,data$$

5.4 Simulations of EEEP Strategy on San Diego Series

After the choice of all the parameters for the correct working of the algorithm, simulations with the software MATLAB can start. The trace considered has 300000 data which represent the number of the transmitted packets every millisecond with the corresponding bits. In the simulations only 200000 samples are considered because MATLAB is not able to handle great amount of data. The following figures show the behavior of the algorithm during the packets transmission, focusing on a block which uses the prediction strategy previously described.

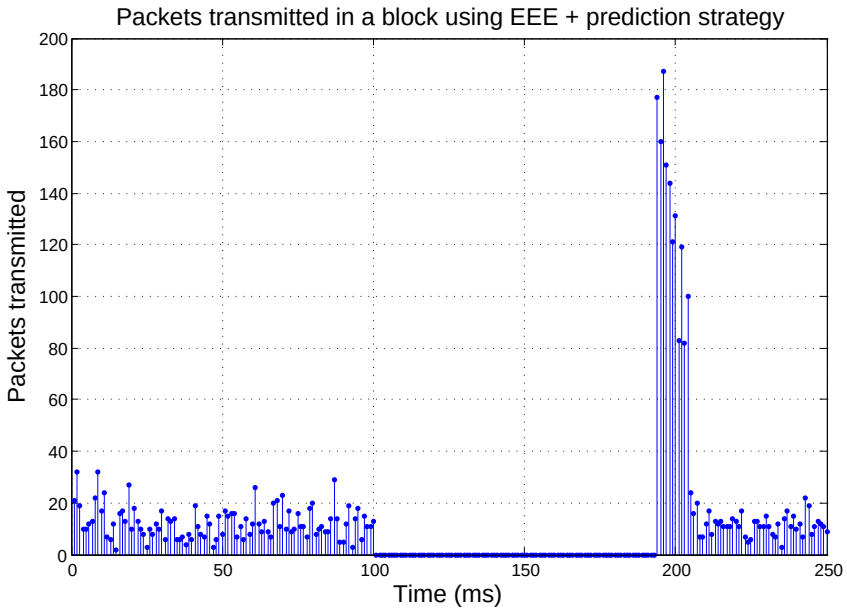

Figure 5.8: *Example of a block which uses EEEP strategy.*

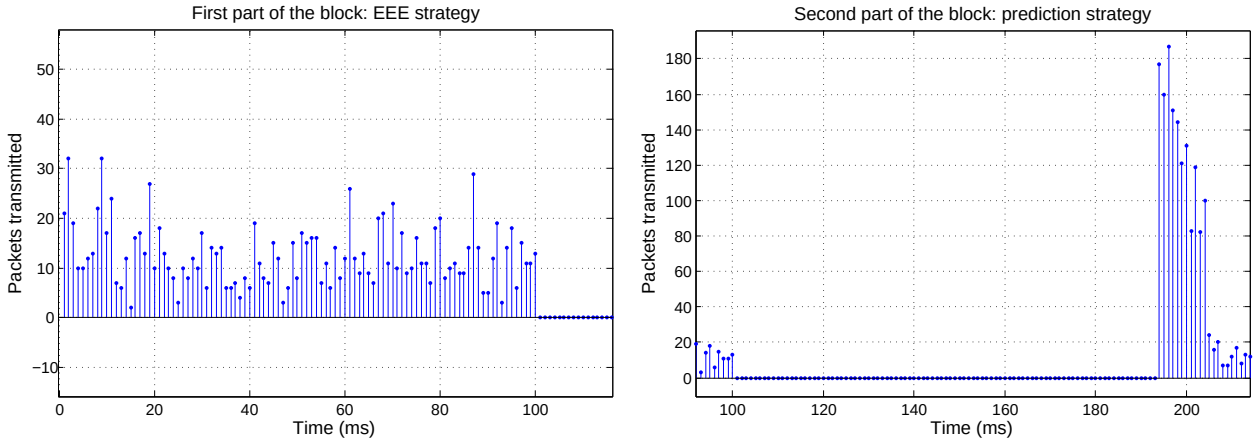

Figure 5.9: *Example of a block which uses EEEP strategy. First part of the block (on the left), second part of the block (on the right).*

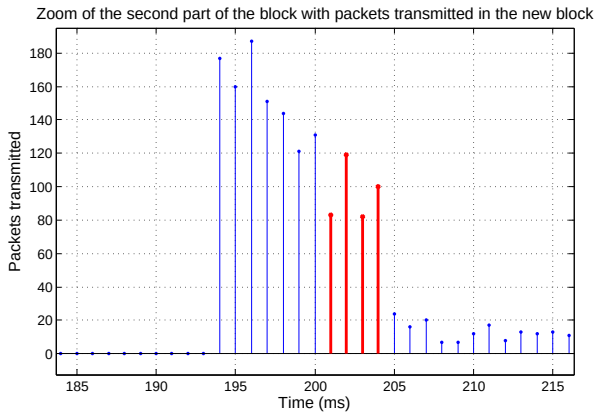

Figure 5.10: *Example of a block which uses EEEP strategy. Zoom of the second part of the block.*

As it can be observed from Fig.5.9, the block of length 200 ms is divided into two parts: the first part of length 100 ms only uses the EEE strategy (every millisecond the link turns on, transmits the packets in the buffer, turns off and wait for the next millisecond), while in the second part (100 ms) the link is off and does not transmit packets until just before the end of the block when it turns on and start to transmit the packets accumulated in the queue (to note the great amount of transmitted packets in the last milliseconds before the end of the block at 200 ms). In Fig.5.10 it is interesting to observe that the block is not able to transmit all the packets accumulated in the queue by the end of the block and it has the necessity to transmit the remaining packets in the new one (in red the milliseconds used in the

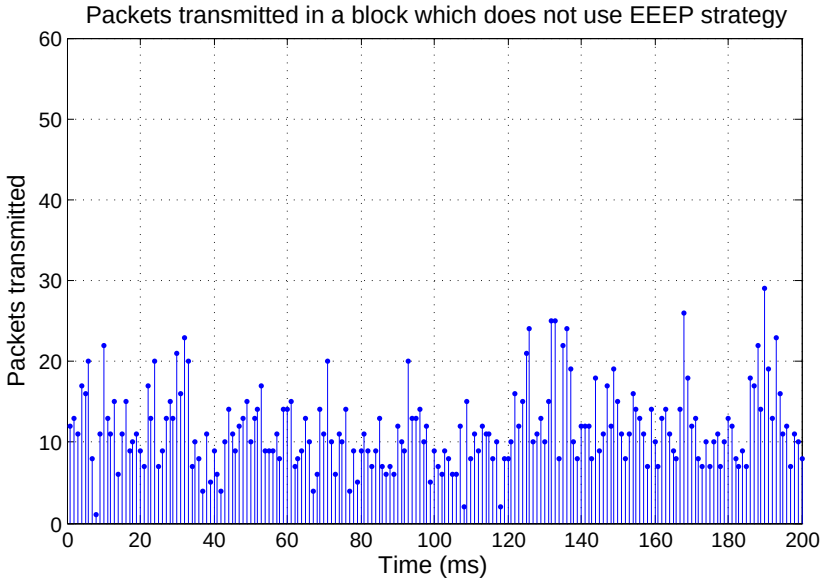

Figure 5.11: *Example of a block which does not use EEEP strategy.*

new block to transmit the remaining packets of the queue). Finally in Fig.5.11 it is reported a block that does not use prediction strategy but it only uses EEE strategy.

In Fig.5.12 the trend of the Hurst parameter, computed every ND data (starting from 20000 data) on windows of LW data, is reported.

Figure 5.12: *Hurst parameter trend.*

Finally, the most interesting figures are those about time and energy performance of the whole algorithm. For a comparison in these figures also two other functions are reported: there is the function which represents the link always on (no efficiency strategy) and the function which represents the link which only uses the EEE strategy.

Figures 5.13, 5.14, 5.15, 5.16 report these functions with their zooms.

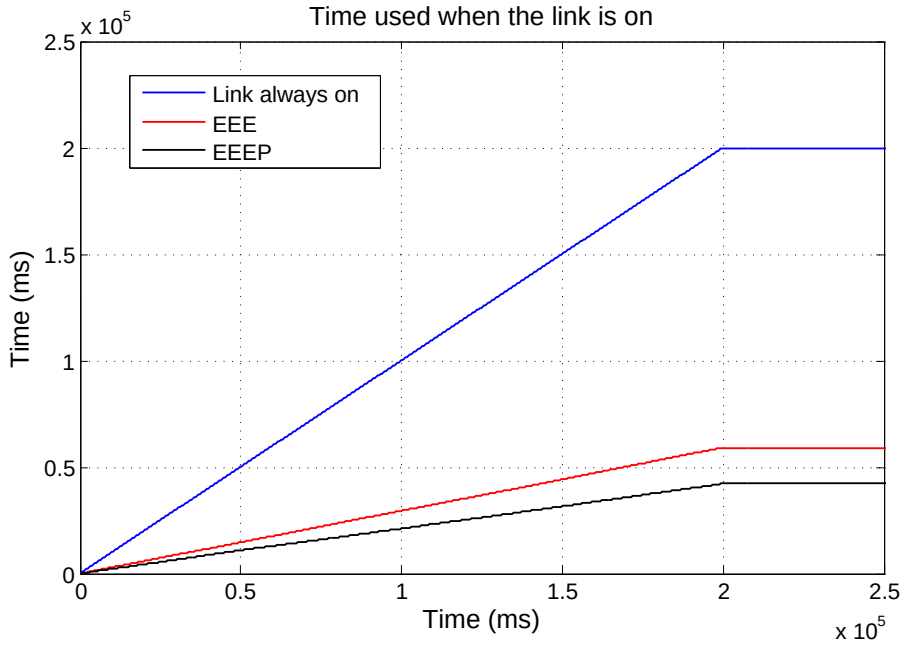

Figure 5.13: *Time spent when the link is on.*

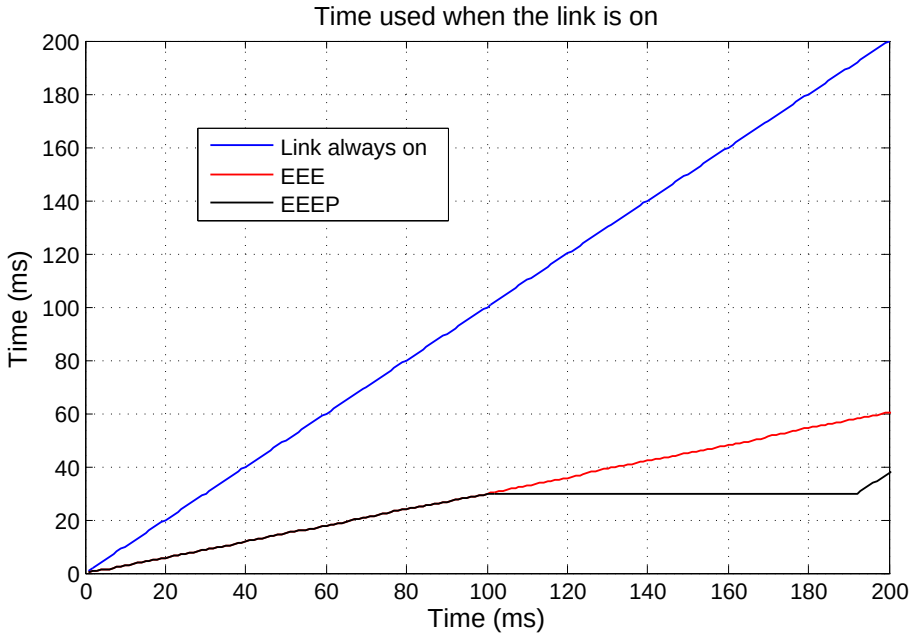

Figure 5.14: *Zoom of the time spent when the link is on.*

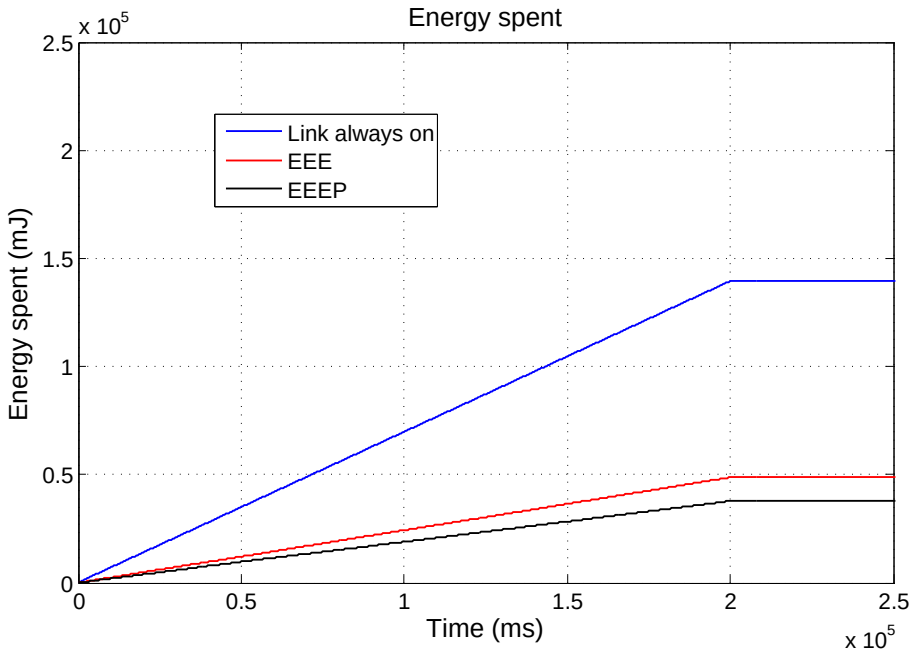

Figure 5.15: *Energy spent when the link is on.*

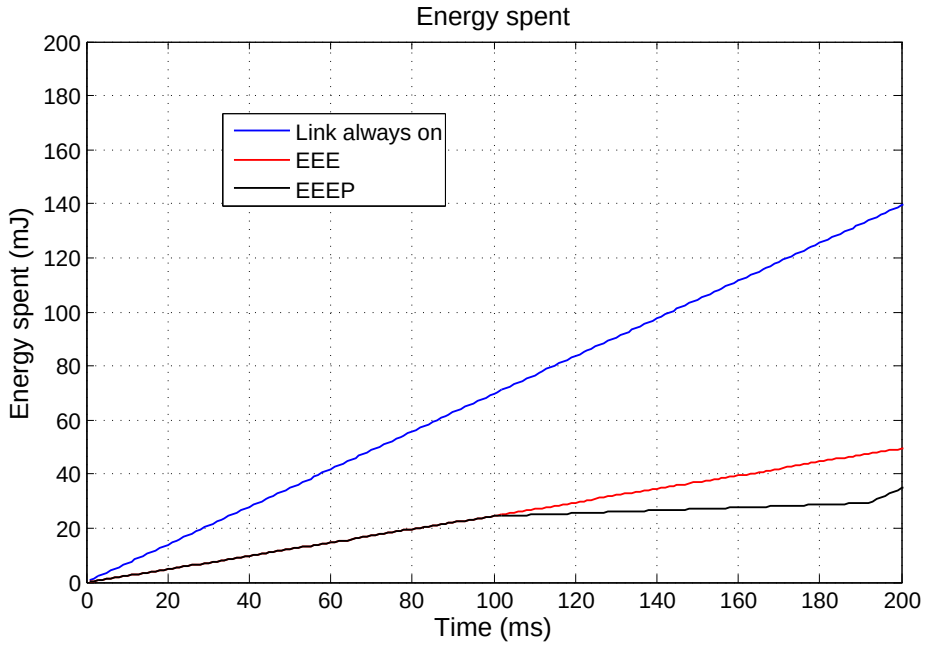

Figure 5.16: *Zoom of the energy spent when the link is on.*

In Fig.5.13 there is the representation of the spent time by the link to stay on. The blue line is the case of the link always on: after 200000 ms = 200 s, the link stayed on for 200 s. The red line refers to the EEE case where the link stays on only for 59.16 s with a time gain of 70.4 % respect the "always on" case. Finally, the black line is that one which represents the algorithm above described and here the link, after 200 s, stays on only for 42.23 s with a time gain of 78.9 % respect to the "always on" case and of 28.6 % respect to the EEE case.

Fig.5.14 is particularly meaningful. In fact, it can be observed the time spent relatively to the first block (it is a zoom of Fig.5.13). The presented algorithm shows its benefits when the link, after the first part of the block, turns off and avoids all the time which EEE strategy spends on transitions (t_w and t_s). After 200 ms, the link using the EEE strategy spends 60.31 ms on while it only spends 37.69 ms using the EEEP strategy with a gain of 37.5 % respect to EEE case.

The same considerations can be done for the two figures about the spent energy by the link when it is on. It is supposed to use a particular Intel card where the power consumption is 0.697 W when the link is on (also considering the transitions) and 0.053 W when it is off. Fig.5.15 shows that, after 200 seconds, if the link is always on, the spent energy is 139.4 J; if the link adopts EEE strategy, the spent energy is 48.7 J with a gain of 65.1 % respect to the "always on" case; if the link adopts EEE + Prediction strategy the spent energy is 37.8 J with a gain of 72.9 % respect to the "always on" case and a gain of 22.4 % respect to the EEE strategy.

Fig.5.16 shows the energy trend for the three cases. After 200 ms, the link "always on" spends 139.4 mJ, the link using EEE strategy spends 49.44 mJ and the link adopting EEEP strategy spends 34.88 mJ with a gain of 29.5 % respect to the EEE case. As it can be seen in particular in this figure, energy gains are lower than the time gains because the great benefits of the algorithm come from the fact that the link stays completely off for a long interval in the block. For the energy case, when the link is off, anyway it spends a few tens of milliWatt (not zero) and for this reason the benefits are lower than the time case.

In conclusion, EEE strategy is very useful to reduce energy consumptions and, combined with the use of the prediction of the future traffic, can become even more effective with more time and energy savings.

For completeness in Tab.5.4 some data about the simulation with the window of 100 ms are reported.

Block #	Packets arrived in the first part	Packets trans. in the first part	Useful time for trans.	Packets trans. in the second part	Packets not trans.	Time for trans.
1	1450	1450	8	1367	164	2
2	1363	1527	9	1337	0	0
3	1206	1206	8	1113	0	0
4	1108	1108	/	1228	0	0
5	1194	1194	8	1140	0	0
6	1107	1107	/	1231	0	0
7	1223	1223	7	1174	0	0
8	1187	1187	6	1149	72	1
9	1121	1193	/	1265	0	0
10	1338	1338	8	1344	0	0
11	1445	1445	9	1270	0	0
12	1233	1233	7	1187	57	2
13	1219	1276	9	1379	0	0
14	1498	1498	10	1407	0	0
15	1424	1424	9	1278	0	0
16	1250	1250	7	1256	0	0
17	1173	1173	7	1144	116	2
18	1271	1387	9	1322	0	0
19	1168	1168	/	1220	0	0
20	1217	1217	7	1141	23	1
21	1117	1140	/	1225	0	0
22	1149	1149	/	1335	0	0

Table 5.4: *Analysis of the first 22 blocks.*

Each row of the table represents a block of traffic of 200 ms. Consecutive rows correspond to consecutive blocks. This table has six columns: in the first one there are the packets arrived in the first 100 ms of every block and in the second one the transmitted packets in the same 100 ms with EEE policy. It is interesting to note the in some blocks the transmitted packets are more than the arrived packets (for

example the second block): this is because in the previous block there are some packets (whose number is reported in the fifth column) that the link is not able to transmit in time; in fact, if the useful time calculated and reported in the third column is not sufficient for the transmission of the packets arrived in the second part, these packets are transmitted in the new subsequent block with a required time reported in the sixth column. The fourth column reports the number of packets which the link is able to transmit in the second 100 ms of every block. The symbol "/" which sometimes appears in the third column means that the prediction strategy is not applied in the following 100 ms of that block.

To conclude this analysis the following data are reported: of 1000 blocks of length 200 ms, 815 used the prediction strategy (81.5 %) and of these 815 blocks, 208 had packets which had to be transmitted in the subsequent block (25.5 %). Then, the maximum number of these packets is 578 and the maximum time used in the subsequent block to transmit these packets is 6 ms (use of 6 % of the window of 100 ms).

Of course one could adopt a more aggressive strategy without allowing any useful time extension and this would lead to higher savings at the cost of possible higher delays (in this case more blocks would exceed their reserved time for transmission).

Finally, in Appendix A the results of other simulations are reported with a final discussion on the computational complexity of the proposed algorithm while in Appendix B MATLAB code is reported.

Conclusions

To sum up the structure of this thesis, it can be divided in four parts.

The first one discusses some models for data networks (Chapter 1) and focuses, in particular, on the model of the self-similarity describing all the mathematical properties, with special attention towards the correlation structure among samples very far (long range dependence) which permits to make a prevision of the future traffic with a certain reliability depending on the value of the Hurst parameter (Chapter 2).

The second part of this thesis concerns the more recent study of energy savings, also in the field of data transmission in a LAN and analyses the recent introduction of the standard IEEE 802.3az and, in particular, the LPI mode, which will be used for the implementation of the algorithm proposed.

The third and the fourth part represent the original contribution and thus the more interesting. The third part at the beginning explains some control tools used while at the end deals with the definition of the control strategy proposed which collects all the elements previously described with a final flow chart which summarizes the strategy.

The fourth part concerns the simulations made with the software MATLAB in which the strategy proposed is analyzed in all its details. Basically it can be established that the use of EEE strategy brings great energy savings with a gain of more than the 70 % respect the case of the link always on. Further gains can be obtained exploiting the statistical properties of the traffic trace using the prediction. With the use of this powerful approach energy savings increase: it can be said that this thesis proposes a more energy efficient technique with interesting results that can be implemented on a real Ethernet card.

Appendix A

Other simulations

Windows Sizes T_1 (ms)	Increments of useful time (%)	Blocks with useful time not sufficient (%)	Time gains (%)	Energy gains (%)
50 ms	+ 0	47.70	34.87	27.2
(2000 blocks)	+ 10	27.01	33.62	26.22
93.5 % of blocks	+ 20	12.78	32.41	25.28
use prediction	+ 30	4.76	31.24	24.37
	+ 40	1.76	30.03	23.43
	+ 50	0.53	28.86	22.51
	+ 60	0.16	27.62	21.55
	+ 70	0.05	26.42	20.61
	+ 80	0.05	25.25	19.70
	+ 90	0	24.04	18.75
100 ms	+ 0	49.35	34.73	27.09
(1000 blocks)	+ 10	21.61	33.51	26.14
93 % of blocks	+ 20	8.82	32.32	25.21
use prediction	+ 30	2.37	31.15	24.30
	+ 40	0.65	29.93	23.35
	+ 50	0.22	28.78	22.45
	+ 60	0.11	27.58	21.52
	+ 70	0	26.36	20.57

Table A.1: *Windows from 50ms to 100ms*

Windows Sizes T_1 (ms)	Increments of useful time (%)	Blocks with useful time not sufficient (%)	Time gains (%)	Energy gains (%)
200 ms	+ 0	48.72	35.07	27.36
(500 blocks)	+ 10	19.36	33.88	26.43
94 % of blocks	+ 20	6.38	32.67	25.49
use prediction	+ 30	0.64	31.48	24.56
	+ 40	0	30.28	23.62
500 ms	+ 0	45.66	32.29	25.19
(200 blocks)	+ 10	13.29	31.18	24.32
86.5 % of blocks	+ 20	4.62	30.06	23.45
use prediction	+ 30	0	28.97	22.60
1000 ms	+ 0	47	37.42	29.20
(100 blocks)	+ 10	15	36.17	28.22
100 % of blocks	+ 20	3	34.91	27.23
use prediction	+ 30	0	33.65	26.25
2000 ms	+ 0	54.76	31.35	24.46
(50 blocks)	+ 10	11.90	30.27	23.62
84 % of blocks	+ 20	2.38	29.20	22.78
use prediction	+ 30	0	28.11	21.93
2500 ms	+ 0	65.62	29.98	23.39
(40 blocks)	+ 10	12.5	28.95	22.57
80 % of blocks	+ 20	6.25	27.93	21.79
use prediction	+30	0	26.91	20.99
5000 ms	+ 0	50	33.50	26.13
(20 blocks)	+ 10	11.11	32.35	25.24
90 % of blocks	+ 20	0	31.20	24.34
use prediction				

Table A.2: *Windows from 200ms to 5000ms*

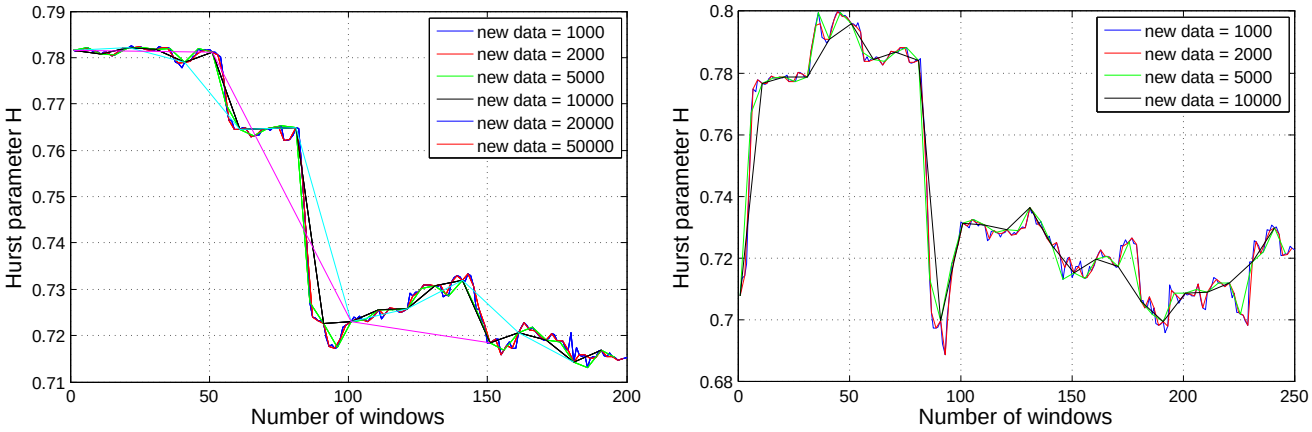

Figure A.1: *Window length = 100000 data (on the left) and 50000 data (on the right)*

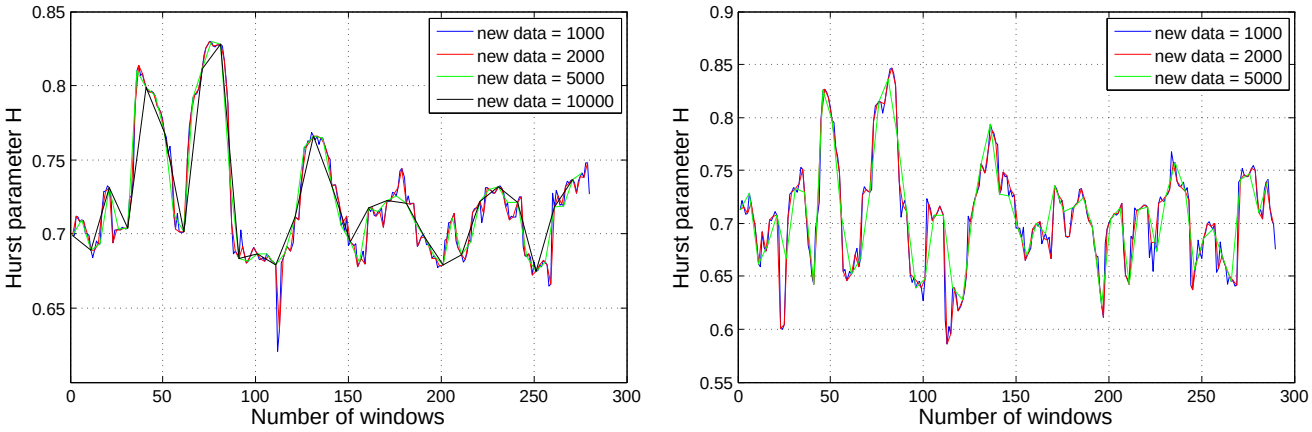

Figure A.2: *Window length = 20000 data (on the left) and 10000 data (on the right)*

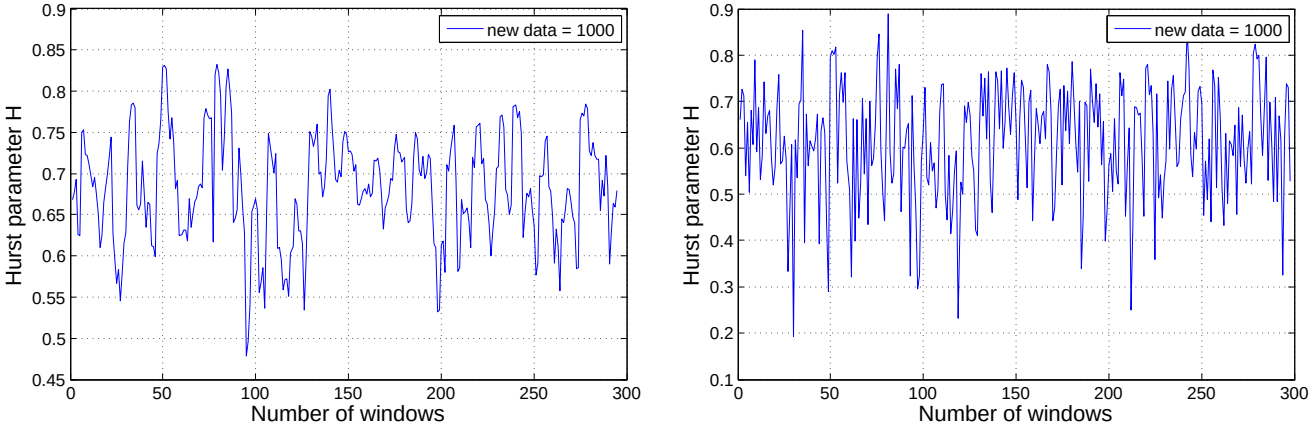

Figure A.3: *Window length = 5000 data (on the left) and 2000 data (on the right)*

The previous tables and figures are those ones obtained analyzing another trace of traffic respect that one reported in Chapter 5. The tables, as already explained, are useful to understand the behavior of the algorithm choosing different values for T_1 while the figures show the trend of the Hurst parameter with different values for LW and ND. The considerations previously done in Chapter 5 are the same and therefore here will not be reported.

In the following table, instead, there are some other data which give an idea of the working of the algorithm in all the windows considered. There are six columns: the first one contains the different values for T_1 chosen. The second one reports the average number of packets which have to be transmitted in the subsequent block and in the third the average time used to transmit these packets. In the fourth one there are also the percentages of time used in a window of length T_1 to transmit the packets in the second column. Finally in the last two ones there are the time and energy gains respect the EEE policy. It can be noted that the gains are also high although windows sizes change.

Windows Size T_1	Average packets trans. in the follow. block	Average time for trans. in the follow. block	Perc. of window used	Time gain %	Energy gain %
50 ms	63	1.57 ms	3.1 %	26.92 %	21.06 %
100 ms	116	2.04 ms	2 %	27.86 %	21.8 %
200 ms	204	2.79 ms	1.4 %	27.75 %	21.7 %
500 ms	466	5.26 ms	1.1 %	25.48 %	19.94 %
1000 ms	977	9.95 ms	1 %	25.8 %	20.19 %
2000 ms	1957	18.42 ms	0.9 %	32.47 %	25.41 %
2500 ms	1990	18.69 ms	0.8 %	20.77 %	16.25 %
5000 ms	3169	29.43 ms	0.6 %	32.45 %	25.39 %

Table A.3: *Different windows simulations.*

Another interesting analysis concerns the computational burden of the proposed algorithm. For computing the time that the software MATLAB used in the various simulations, the commands *tic* and *toc* were used. In the following table there are some time measured varying the window size T_1 and the parameters LW and ND.

Size T_1	LW	ND	Time
50 ms	20000	1000	5.94 s
	100000	10000	5.06 s
	100000	50000	4.77 s
100 ms	20000	1000	5.43 s
	100000	10000	4.52 s
	100000	50000	4.25 s
200 ms	20000	1000	5.19 s
	100000	10000	4.22 s
	100000	50000	4.16 s
500 ms	20000	1000	4.83 s
	100000	10000	3.92 s
	100000	50000	3.67 s
1000 ms	20000	1000	4.77 s
	100000	10000	3.93 s
	100000	50000	3.69 s
2000 ms	20000	1000	4.99 s
	100000	10000	4.11 s
	100000	50000	3.83 s
5000 ms	20000	1000	5.01 s
	100000	10000	4.04 s
	100000	50000	3.78 s

Table A.4: *Time taken by the algorithm to work in MATLAB.*

As it can be noted, the time taken by the algorithm to work seems to decrease as the length of the window increases. Another interesting thing is that, fixing the window, using different values of LW and ND, the computational time decreases of more than 1 second with the increase of LW and ND. This fact is easy to

understand: in fact, with longer windows and with more new data considered, a smaller number of times there is the Hurst parameter computation and this brings a great time savings.

In Fig.A.4 there is the graphical representations of the computational burden of the algorithm when it runs in MATLAB.

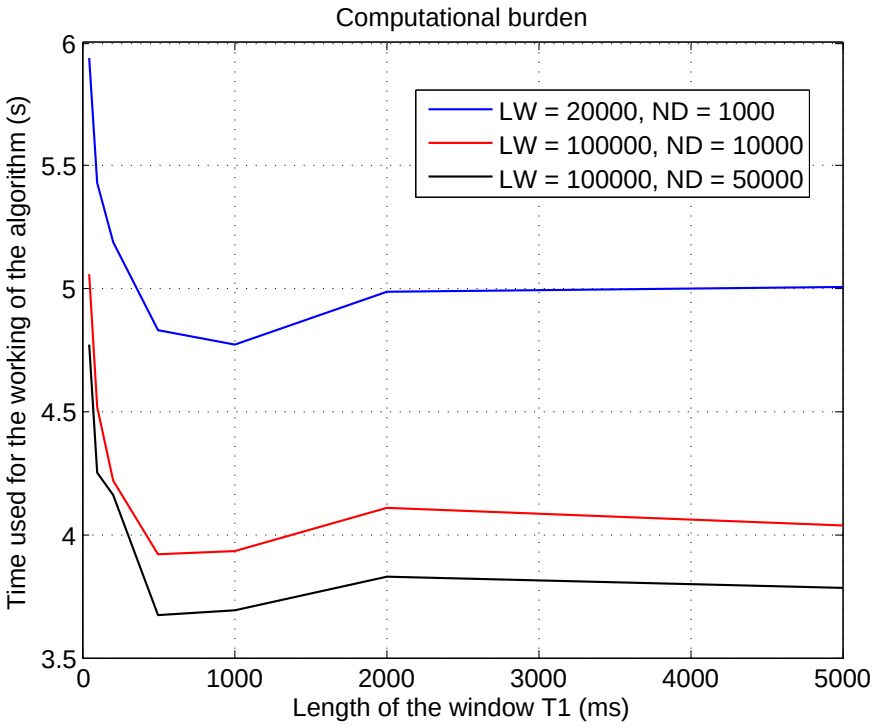

Figure A.4: *Graphical representation of the computational burden of the algorithm.*

Appendix B

Matlab Code

In this appendix the implementation of the algorithm proposed using the software MATLAB is reported.

```
clear all
clc
close all

%% data

% inputs
finestra = 100;
LW = 20000;
ND = 1000;
f = 10^6; % bit/millisec (1 Gbit/s, 1 bit/nanosec)
perc = 0;
tsleep = 0.202;
twake = 0.0165;
ttrans = tsleep + twake;
enactive = 0.697; % Watt spent when link is on + transistions
enquiet = 0.053;  % Watt spent when link is off

% open the file
fid = fopen('20040130-1-1-T1ms.txt','r');

% save in v
```

```
23 v = fscanf(fid,'%d',[2 inf]);
24
25 %close the file
26 fclose(fid);
27
28 % total data
29 tot = 300000;
30
31 % data considered
32 N = 200000;
33
34 % packets vector
35
36 p = zeros(65,tot);
37
38 p(1,:) = v(2,:);
39
40 % bits vector
41 bit = v(1,:)*8;
42
43 % packets counters
44 nump = 2;
45 numpac = 2;
46
47 % transmitted packets
48 ptrasm = zeros(1,length(p(1,1:N)));
49
50 % useful time variables
51 tpass = 0;
52 tpasspred = 0;
53 tp = 0;
54 tp1 = 0;
55 tp2 = 0;
56
57 % wasted time in the window
58 waste = finestra*ttrans;
59
60 % transmission times vector
```

```
61 T = zeros(1,N);
62
63 % Traffic volume
64 V = 0;
65
66 % window index
67 ind = 1;
68
69 % blocks indices
70 l = 1;
71 j = 0;
72
73 % times vector when the link turns on
74 sicurezza = zeros(1,N/(2*finestra));
75
76 % traffic levels in the first part of the blocks
77 L1 = zeros(1,N/(2*finestra));
78
79 % useful boolean variables
80 pred = false;
81 fine = false;
82
83 % queue variables
84 cod = 1;
85 row = 1;
86 coda = zeros(1,100000);
87
88 % variables for Hurst parameter control
89
90 ts = 1;
91 Hu = zeros(1,(N-LW)/ND + 2);
92 Hu(1) = 0.7;
93 ph = zeros(7,LW);
94 varianza = zeros(1,7);
95 agg = [5,10,50,100,500,1000];
96 agg1 = [1,agg];
97 logagg1 = log10(agg1);
98
```

```
99  %% table construction
100
101 % maximum and minimum traffic
102
103 Vmin = sum(p(1,1:finestra));
104 Vmax = sum(p(1,1:finestra));
105
106 for c = 1:finestra:(tot - finestra)
107     Vol = sum(p(1,c:c+finestra-1));
108     if Vol > Vmax
109         Vmax = Vol;
110     end
111     if Vol < Vmin
112         Vmin = Vol;
113     end
114 end
115
116 % quantization levels
117
118 d = 8;
119 mu = (Vmax - Vmin)/d;
120 primaparte = true;
121 e = 1;
122 blocks = zeros(d,d+1);
123
124 Liv1 = zeros(1,tot/(2*finestra));
125 Liv2 = zeros(1,tot/(2*finestra));
126 luno = 1;
127
128 % table building
129
130 for g = 1:finestra:(tot - finestra + 1)
131
132     Vol = sum(p(1,g:g+finestra-1));
133
134     if(primaparte)
135      if Vol < Vmin + mu
136         Liv1(luno) = 1;
```

```
137         end
138         if Vol >= Vmin + mu && Vol < Vmin + 2*mu
139            Liv1(luno) = 2;
140         end
141         if Vol >= Vmin + 2*mu && Vol < Vmin + 3*mu
142            Liv1(luno) = 3;
143         end
144         if Vol >= Vmin + 3*mu && Vol < Vmin + 4*mu
145            Liv1(luno) = 4;
146         end
147         if Vol >= Vmin + 4*mu && Vol < Vmin + 5*mu
148            Liv1(luno) = 5;
149         end
150         if Vol >= Vmin + 5*mu && Vol < Vmin + 6*mu
151            Liv1(luno) = 6;
152         end
153         if Vol >= Vmin + 6*mu && Vol < Vmin + 7*mu
154            Liv1(luno) = 7;
155         end
156         if Vol >= Vmin + 7*mu
157            Liv1(luno) = 8;
158         end
159
160         blocks(Liv1(luno),9) = blocks(Liv1(luno),9) + 1;
161
162       else
163
164         if Vol < Vmin + mu
165            Liv2(luno) = 1;
166         end
167         if Vol >= Vmin + mu && Vol < Vmin + 2*mu
168            Liv2(luno) = 2;
169         end
170         if Vol >= Vmin + 2*mu && Vol < Vmin + 3*mu
171            Liv2(luno) = 3;
172         end
173         if Vol >= Vmin + 3*mu && Vol < Vmin + 4*mu
174            Liv2(luno) = 4;
```

```
175       end
176       if Vol >= Vmin + 4*mu && Vol < Vmin + 5*mu
177          Liv2(luno) = 5;
178       end
179       if Vol >= Vmin + 5*mu && Vol < Vmin + 6*mu
180          Liv2(luno) = 6;
181       end
182       if Vol >= Vmin + 6*mu && Vol < Vmin + 7*mu
183          Liv2(luno) = 7;
184       end
185       if Vol >= Vmin + 7*mu
186          Liv2(luno) = 8;
187       end
188
189       blocks(Liv1(luno),Liv2(luno)) = blocks(Liv1(luno),Liv2(
              luno)) + 1;
190
191      end
192
193      e = e + 1;
194
195      if (mod(e,2) == 0)
196          primaparte = false;
197      else
198          primaparte = true;
199          luno = luno + 1;
200      end
201
202  end
203
204  % conditional probability matrix
205
206  CondProb = blocks(:,1:8);
207
208  for q = 1:8
209      if blocks(q,9) == 0
210          CondProb(q,:) = zeros(1,8);
211          continue;
```

```
212     end
213     CondProb(q,:) = CondProb(q,:)./blocks(q,9);
214 end
215 
216 % trend of the conditional probability densities
217 
218 figure(1)
219 for u = 1:8
220     hold on
221     plot3(u*ones(1,8),1:8,CondProb(u,1:8))
222     grid on
223 end
224 xlabel('Traffic level L1','FontSize',14)
225 ylabel('Traffic level L2','FontSize',14)
226 zlabel('P(L2|L1 = l)','FontSize',14)
227 title('Conditional probability densities','FontSize',14)
228 
229 
230 medio = zeros(1,8);
231 
232 % average value for each line
233 
234 for t = 1:8
235     for i = 1:8
236         medio(t) = medio(t) + i*CondProb(t,i);
237     end
238 end
239 
240 %% bits division in the packets
241 
242 cou = 1;
243 for t=1:N
244     while(cou < p(1,t)+1)
245         if cou == p(1,t)
246            p(cou+1,t) = bit(t) - sum(p(2:cou,t));
247            break
248         end
249         p(cou+1,t) = round(bit(t)/p(1,t));
```

```
        cou = cou + 1;
    end
    cou = 1;
end


%% packets transmissions

for t = 1:N

 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % H COMPUTATION

  % Hurst parameter control
  if t == LW + (ts-1)*ND

      % data used for H computation
      ph(1,:) = p(1, 1 + (ts-1)*ND : LW + (ts-1)*ND);

      % partial sum
      y = 0;

      % row index
      z = 1;

      % computation
      for h = 1 : length(agg)
          for w = 1 : LW / agg(h)
              for k = 1+(w-1)*agg(h) : agg(h)+(w-1)*agg(h)
                  y = y + ph(1,k);
              end
              ph(z+1,w) = y / agg(h);
              y = 0;
          end
          z = z + 1;
      end
```

```
286
287         % variances of the aggregate series
288         for n = 1 : length(agg1)
289             varianza(n) = var(ph(n, 1 : LW/agg1(n)));
290         end
291
292         logvarianza = log10(varianza);
293
294         % least squares line
295         coeff = polyfit(logagg1,logvarianza,1);
296         Hu(ts+1) = 1 + coeff(1)/2;
297
298         ts = ts + 1;
299
300     end
301
302 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
303       % EEE + PREDICTION STRATEGY
304
305       % queue building for the packets arrived when the link is
              off
306
307       if Hu(ts) > 0.6 && pred == true && 2*j*finestra -
              finestra < t && t <= 2*j*finestra - sicurezza(j)
308           ptrasm(t) = 0;
309           count = p(1,t);
310           if p(1,t) == 0
311              coda(1,cod) = 0;
312              cod = cod + 1;
313           end
314           while(count > 0)
315               coda(1,cod) = p(count+1,t);
316               count = count - 1;
317               cod = cod + 1;
318           end
319           continue;
320       end
321
```

```
322     % when the link turns on, there is the transmission of
            the packets
323     % arrived at instant t and in the queue
324
325     if Hu(ts) > 0.6 && pred == true && 2*j*finestra -
            sicurezza(j) < t && t <= 2*j*finestra
326
327         while(tp < 1)
328
329             if p(2,t) == 0
330                 break
331             end
332
333             % immediate transmissions of the packets which
                    arrive at
334             % instant t
335             ptrasm(t) = ptrasm(t) + 1;
336             tp = tp + p(numpac,t)/f;
337             numpac = numpac + 1;
338
339             % start with the transmissions of packets in the
                    queue
340             if numpac == p(1,t)+2
341
342               T(t) = T(t) + tp;
343
344               while(tpasspred < 1 - tp)
345
346                if (tpasspred + coda(row)/f < 1 - tp)
347                    ptrasm(t) = ptrasm(t) + 1;
348                    tpasspred = tpasspred + coda(row)/f;
349                    row = row + 1;
350
351                    % queue ends
352                    if row == cod
353                        fine = true;
354                        pred = false;
355                        row = 1;
```

```
356                         cod = 1;
357
358                         coda = zeros(1,100000);
359                         tpasspred = 0;
360                          break
361                     end
362                 else
363                      T(t) = T(t) + tpasspred;
364                      tpasspred = 0;
365                     break
366                 end
367                end
368                break
369             end
370         end
371         numpac = 2;
372         tp = 0;
373         continue
374     end
375
376     % when queue ends normal transmission
377
378     if Hu(ts) > 0.6 && fine == true && 2*j*finestra -
          sicurezza(j) < t && t <= 2*j*finestra
379
380         while(tp1 < 1)
381
382             if p(2,t) == 0
383                break
384             end
385
386             ptrasm(t) = ptrasm(t) + 1;
387             tp1 = tp1 + p(numpac,t)/f;
388             numpac = numpac + 1;
389
390             if numpac == p(1,t)+2
391                 T(t) = T(t) + tp1;
392                 tp1 = 0;
```

```
393                 numpac = 2;
394                 break
395             end
396         end
397
398         if t == 2*j*finestra
399             fine = false;
400         end
401         continue
402     end
403
404 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
405     % EEE STRATEGY
406
407     while(tpass < 1 - ttrans)
408
409         if p(2,t) == 0
410            break
411         end
412
413         ptrasm(t) = ptrasm(t) + 1;
414         tpass = tpass + p(nump,t)/f;
415         nump = nump + 1;
416
417         if nump == p(1,t)+2
418
419            T(t) = tpass + ttrans;
420
421            % counter for volume traffic arrived up to t
422            V = V + ptrasm(t);
423
424            % if the queue is not empty
425            if row < cod
426                while (tp2 < 1 - T(t))
427
428                    if (tp2 + coda(row)/f < 1 - T(t))
429
430                        ptrasm(t) = ptrasm(t) + 1;
```

```
431                     tp2 = tp2 + coda(row)/f;
432                     row = row + 1;
433                     if row == cod
434                        row = 1;
435                        cod = 1;
436
437                        coda = zeros(1,100000);
438                        break
439                     end
440                 else
441                     T(t) = T(t) + tp2;
442                     tp2 = 0;
443                     break
444                 end
445             end
446         end
447
448         tpass = 0;
449         nump = 2;
450
451         % when the first part of the block ends, the
              volume traffic is
452         % assigned to a quantization level and there is
              the decision if
453         % it is appropriate to apply the prediction
              strategy
454
455         if ind == finestra
456             if V < Vmin + mu
457                L1(l) = 1;
458             end
459             if V >= Vmin + mu && V < Vmin + 2*mu
460                L1(l) = 2;
461             end
462             if V >= Vmin + 2*mu && V < Vmin + 3*mu
463                L1(l) = 3;
464             end
465             if V >= Vmin + 3*mu && V < Vmin + 4*mu
```

```
466                 L1(l) = 4;
467             end
468             if V >= Vmin + 4*mu && V < Vmin + 5*mu
469                 L1(l) = 5;
470             end
471             if V >= Vmin + 5*mu && V < Vmin + 6*mu
472                 L1(l) = 6;
473             end
474             if V >= Vmin + 6*mu && V < Vmin + 7*mu
475                 L1(l) = 7;
476             end
477             if V >= Vmin + 7*mu
478                 L1(l) = 8;
479             end
480
481             % if in the second part of the block there is
                    less traffic or
482             % equal traffic respect to the first part of
                    the block use of
483             % the prediction strategy
484
485             if L1(l) >= round(medio((L1(l))))
486                 tempoL1 = 0;
487                 for i = 1 + 2*(l-1)*finestra : finestra +
                        2*(l-1)*finestra
488                     tempoL1 = tempoL1 + T(i);
489                 end
490
491                 % useful time
492                 useful = tempoL1 - waste;
493
494                 % there is the possibility to add a
                        percentage to the
495                 % useful time
496
497                 sicurezza(l) = useful + perc*useful/100;
498                 sicurezza(l) = ceil(sicurezza(l));
499                 pred = true;
```

```
500
501                     ind = 0;
502                     V = 0;
503
504                 else
505
506                     sicurezza(l) = 0;
507                     pred = false;
508                     ind = - finestra;
509
510                 end
511
512                 l = l + 1;
513                 j = j + 1;
514
515             end
516             break;
517         end
518     end
519
520     ind = ind + 1;
521     if ind == 0
522        V = 0;
523     end
524
525 end
526
527 %% Time analysis
528
529 % time plot using link always on
530
531 tempo1 = zeros(1,2*N);
532 tempo1(1:N) = 1:N;
533 tempo1(N+1:2*N) = N*ones(1,N);
534
535 figure(2)
536 plot(tempo1)
537 axis([0,2*N,0,N+50000])
```

```
538 grid on
539 xlabel('Time (ms)', 'FontSize',14)
540 ylabel('Time (ms)', 'FontSize',14)
541 title('Time used when the link is on', 'FontSize',14)
542 hold on
543
544 % time plot using link with only EEE strategy
545
546 Teff = bit/f + ttrans*ones(1,300000);
547 tempo2 = zeros(1,length(T));
548
549 tempo2(1) = Teff(1);
550 for i = 2:length(T)
551     tempo2(i) = tempo2(i-1) + Teff(i);
552 end
553
554 tempo2(N+1:2*N) = tempo2(N)*ones(1,N);
555
556 plot(tempo2,'r')
557
558 hold on
559
560 % time plot using link with EEE + prediction strategy
561
562 tempo3 = zeros(1,length(T));
563
564 tempo3(1) = T(1);
565 for a = 1:(N/(2*finestra))
566     for i = 1 + 2*(a-1)*finestra : finestra + 2*(a-1)*
          finestra - 1
567         tempo3(i+1) = tempo3(i) + T(i+1);
568     end
569     if sicurezza(a) == 0
570        for i = finestra + 2*(a-1)*finestra : 2*a*finestra - 1
571         tempo3(i+1) = tempo3(i) + T(i+1);
572        end
573        if 2*a*finestra + 1 == N + 1
574           break;
```

```
575         end
576         tempo3(2*a*finestra + 1) = tempo3(2*a*finestra) + T(2*
                a*finestra + 1);
577         continue
578     end
579     tempo3(finestra + 1 + 2*(a-1)*finestra : 2*finestra + 2*(
            a-1)*finestra - round(sicurezza(a))) = tempo3(
            finestra + 2*(a-1)*finestra)*ones(1,finestra-round(
            sicurezza(a)));
580     tempo3(2*finestra + 2*(a-1)*finestra - round(sicurezza(a)
            ) + 1 : 2*finestra + 2*(a-1)*finestra) = tempo3(2*
            finestra + 2*(a-1)*finestra - round(sicurezza(a))) +
            (1:round(sicurezza(a)));
581     if 2*a*finestra + 1 == N + 1
582         break;
583     end
584     tempo3(2*a*finestra + 1) = tempo3(2*a*finestra) + T(2*a*
            finestra + 1);
585 end
586
587 tempo3(N+1:2*N) = tempo3(N)*ones(1,N);
588
589 plot(tempo3,'k')
590 grid on
591
592 legend('Link always on','EEE','EEE + Prediction')
593
594 %% Energy analysis (Intel card)
595
596 % energy plot using link always on
597
598 energia1 = zeros(1,2*N);
599 energia1(1:N) = enactive*(1:N);
600 energia1(N+1:2*N) = energia1(N)*ones(1,N);
601 figure(3)
602 plot(energia1)
603 axis([0,2*N,0,N+50000])
604 grid on
```

```
605 xlabel('Time (ms)', 'FontSize',14)
606 ylabel('Energy spent (mJ)', 'FontSize',14)
607 title('Energy spent', 'FontSize',14)
608 hold on
609
610 % energy plot using link with only EEE strategy
611
612 energia2 = (enactive*Teff + enquiet*(ones(1,length(Teff)) -
        Teff));
613 for i = 2:length(T)
614     energia2(i) = energia2(i-1) + energia2(i);
615 end
616
617 energia2(N+1:2*N) = energia2(N)*ones(1,N);
618
619 plot(energia2,'r')
620 hold on
621
622 % energy plot using link with EEE + prediction strategy
623
624 energia3 = zeros(1,length(T));
625
626 energia3(1) = enactive*T(1) + enquiet*(1-T(1));
627 for a = 1:(N/(2*finestra))
628     for i = 1 + 2*(a-1)*finestra : finestra + 2*(a-1)*
          finestra - 1
629         energia3(i+1) = energia3(i) + enactive*T(i+1) +
              enquiet*(1-T(i+1));
630     end
631     if sicurezza(a) == 0
632        for i = finestra + 2*(a-1)*finestra : 2*a*finestra - 1
633         energia3(i+1) = energia3(i) + enactive*T(i+1) +
              enquiet*(1-T(i+1));
634        end
635        if 2*a*finestra + 1 == N + 1
636           break;
637         end
638        energia3(2*a*finestra + 1) = energia3(2*a*finestra) +
```

```
            enactive*T(2*a*finestra + 1) + enquiet*(1-T(2*a*
            finestra + 1));
639         continue
640     end
641     energia3(finestra + 1 + 2*(a-1)*finestra : 2*finestra +
            2*(a-1)*finestra - round(sicurezza(a))) = energia3(
            finestra + 2*(a-1)*finestra) + enquiet*(1 : finestra-
            round(sicurezza(a)));
642     energia3(2*finestra + 2*(a-1)*finestra - round(sicurezza(
            a)) + 1 : 2*finestra + 2*(a-1)*finestra) = energia3
            (2*finestra + 2*(a-1)*finestra - round(sicurezza(a)))
             + enactive*(1   : round(sicurezza(a)));
643     if 2*a*finestra + 1 == N + 1
644         break;
645     end
646     energia3(2*a*finestra + 1) = energia3(2*a*finestra) +
            enactive*T(2*a*finestra + 1) + enquiet*(1-T(2*a*
            finestra + 1));
647 end
648
649 energia3(N+1:2*N) = energia3(N)*ones(1,N);
650
651 plot(energia3,'k')
652 grid on
653
654 legend('Link always on','EEE','EEE + Prediction')
655
656 %% Packets transmitted
657
658 figure(4)
659 stem(ptrasm(118001:118250),'.')
660 grid on
661 xlabel('Time (ms)', 'FontSize',14)
662 ylabel('Packets transmitted', 'FontSize',14)
663 title('Packets transmitted in a block using EEE + prediction
        strategy', 'FontSize',14)
664
665 %%
```

```
666 figure(5)
667 stem(ptrasm(601:800),'.')
668 grid on
669 xlabel('Time (ms)', 'FontSize',14)
670 ylabel('Packets transmitted', 'FontSize',14)
671 title('Packets transmitted in a block using EEE + prediction
        strategy', 'FontSize',14)
672 axis([0,200,0,250])
673
674 %% Hurst parameter
675
676 figure(6)
677 plot(Hu)
678 grid on
679 xlabel('Windows', 'FontSize',14)
680 ylabel('Hurst parameter', 'FontSize',14)
681 title('Trend of the Hurst parameter', 'FontSize',14)
682
683 %% Final analysis
684
685 ana = zeros(N/(2*finestra),10);
686
687 for i = 1:N/(2*finestra)
688     %
689     ana(i,1) = sum(p(1, 1 + 2*(i-1)*finestra : finestra + 2*(
            i-1)*finestra));
690     ana(i,2) = sum(ptrasm(1, 1 + 2*(i-1)*finestra : finestra
            + 2*(i-1)*finestra));
691     ana(i,3) = L1(i);
692     ana(i,4) = medio(L1(i));
693     ana(i,5) = sicurezza(i);
694     ana(i,6) = sum(p(1, 2*i*finestra - finestra + 1 : 2*i*
            finestra - sicurezza(i)));  % in coda
695     ana(i,7) = sum(p(1, 2*i*finestra - sicurezza(i) + 1 : 2*i
            *finestra ));  % arrivano quando è accesa
696     ana(i,8) = sum(ptrasm(1, 2*i*finestra - sicurezza(i) + 1
            : 2*i*finestra ));  % trasmessi
697     ana(i,9) = ana(i,6) +  ana(i,7) -  ana(i,8);  % mancanti
```

```
            da trasmettere
698     if sicurezza(i) == 0
699         ana(i,9) = 0;
700     end
701     if i == N/(2*finestra)
702         break
703     end
704     ana(i,10) = length(find((ptrasm(1, 1 + 2*i*finestra :
            finestra + 2*i*finestra) - p(1, 1 + 2*i*finestra :
            finestra + 2*i*finestra))> 0));
705
706 end
707
708 % gain of time and gain of energy respect to EEE
709 gt = (tempo2(N) - tempo3(N))/tempo2(N)*100
710 ge = (energia2(N) - energia3(N))/energia2(N)*100
711
712 % number of blocks
713 bl = N/(2*finestra)
714
715 % blocks which use prediction strategy
716 pr = length(find(ana(:,5) ~=0 ))
717 pepr = pr/bl*100
718
719 % blocks which have to transmit their packets in the
        following block
720 sf = length(find(ana(:,10) ~=0 ))
721 pesf = sf/pr*100
722
723 % max number of packets transmitted in the following block
        and
724 % max number of time used to transmit old packets in the new
        block
725 pack = find(ana(:,9) > 0);
726 pout = max(ana(pack,9))
727 tco = max(ana(pack,10))
728
729 % percentage of the window used to transmit packets in the
```

```
    new block (max case)
peco = tco/finestra*100
```

Bibliography

[1] Becchi M., *From Poisson Processes to Self-Similarity: a Survey of Network Traffic Models*, Academic Article, December 2006.

[2] Bennett M., Christensen K., Maestro J.A., Mostowfi M., Nordman B., Reviriego P., *IEEE 802.3az: The Road to Energy Efficient Ethernet*, IEEE Communications Magazine, November 2010.

[3] Christensen K., Gunaratne C., Nordman B., Suen S., *Reducing the Energy Consumption of Ethernet with Adaptive Link Rate (ALR)*, IEEE Transactions On Computers, Vol. 57, No. 4, April 2008.

[4] Crovella M., Bestavros A. *Self-similarity in World Wide Web traffic: evidence and possible causes*, Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, May 1996.

[5] Duffield N., O'Connel N., *Large deviations and overflow probabilities for the general single server queue with applications*, Math. Proc. Cambridge Philos. Soc., pp. 363-374, 1995.

[6] Garret M., Willinger W., *Analisys, modeling and generation of self-similar VBR video traffic*, Proceedings ACM SIGCOMM '94, pp.269-280, 1994.

[7] Giovanardi A., *Wireless multimedia systems*, Ph.D. Thesis, February 2000.

[8] Grossglauser M., Bolot J., *On the relevance of long-range dependence in network traffic*, Proc. ACM SIGCOMM '96, pp.15-24, 1996.

[9] Herreria-Alonso S., Rodriguez-Perez M., Fernandez-Veiga M., Lopez-Garcia C., *Optimal configuration of Energy-Efficient Ethernet*, Computer Networks,Vol. 56, pp. 2456-2467, July 2012.

[10] Leland W.E., Taqqu M.S., Willinger W., Wilson D.V., *On the Self-Similar Nature of Ethernet Traffic (Extended Version)*, IEEE/ACM Transactions On Networking, Vol. 2, No. 1, February 1994.

[11] Likhanov N., Tsybakov B., Georganas N., *Analysis of an ATM buffer with self-similar (fractal) input traffic*, IEEE INFOCOM '95, pp. 985-992, 1995.

[12] Mandelbrot B.B., *The Fractal Geometry of Nature*, W.H. Freeman, New York, 1982.

[13] Norros I., *A storage model with self-similar input*, Queueing Systems, pp. 387-396, 1994.

[14] Park K., Willinger W., *Self-Similar Network Traffic and Performance Evaluation*, New York, John Wiley and Sons Inc.,2000.

[15] Park K., Kim G., Crovella M., *On the relationship between file sizes, transport protocols and self-smilar network traffic*, IEEE International Conference on Network Protocols, pp. 171-180, 1996.

[16] Paxson V., Floyd S., *Wide-area traffic: the failure of Poisson modeling*, Proceedings ACM SIGCOMM '94, pp. 257-268, 1994.

[17] Reviriego P., Christensen K., Rabanillo J., Maestro J.A., *An Initial Evaluation of Energy Efficient Ethernet*, IEEE Communications Letters, Vol. 15, No. 5, May 2011.

[18] Taqqu M., Levy J., *Using renewal processes to generate long-range dependence and high variability*, in Progress in Probability and Statistics, Vol. 11, Birkhauser, Boston, 1996.

[19] Tsybakov B., Georganas N., *Self-similar traffic and upper bounds to buffer overflow in an ATM queue*, Performance Evaluation, pp. 57-80, 1998.

[20] Tuan T., Park K., *Multiple time scale congestion control for self-similar netwotk traffic*, Performance Evaluation, pp. 359-386, 1999.

[21] Tuan T., Park K., *Multiple time scale redundancy control for QoS-sensitive transport of real-time traffic*, IEEE INFOCOM '00, 2000.

[22] Willinger W., Taqqu M., Sherman R., Wilson D., *Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level*, Proceedings ACM SIGCOMM '95, pp. 100-113, 1995.

[23] Wilson M., *A Historical View of Network Traffic Models*, Academic Article, December 2006.

[24] Zorzi M., Benvenuto N., *Principles of Communications Networks and Systems*, John Wiley and Sons Ltd, 2010.

[25] *http://en.wikipedia.org/wiki/Hurst exponent*

[26] *http://en.wikipedia.org/wiki/OSI model*

[27] *http://en.wikipedia.org/wiki/Ethernet*